Exhibit I. Glushko, Robert J., Implementing Domain-specific Commerce Languages with a Common Business Library, Slides 29-31 (delivered July 25, 1998) accessed at http://groups.haas.berkeley.edu/citm/conferences/cec/Presentations/Session3/ glushko.pdf on October 26, 2006

Implementing Domain-specific Commerce Languages with a Common Business Library

Dr. Robert Glushko -- Director, Information Engineering

4005 Miranda Avenue, Suite 150

Palo Alto, CA 94304

415.858.7711 main

415.858.4925 fax

www.veosystems.com

info@veosystems.com

R-649

Outline of the Talk

- XML as a technology platform for commerce applications
- Domain-specific commerce languages
- A common business library

About Veo Systems

- **History**
  - 1/97 for-profit "spin-off" of CommerceNet Consortium called "CNgroup"
  - 9/97 received multi-million $ award from U.S. Commerce Department ATP to help commercialize "eCo" component-based commerce framework (along with CommerceNet)
  - 4/98 changed name from CNgroup -Veo
- **Status**
  - Privately held, backed by corporate and VC investors, growing very fast
  - Products later this year

XML as Technology Platform

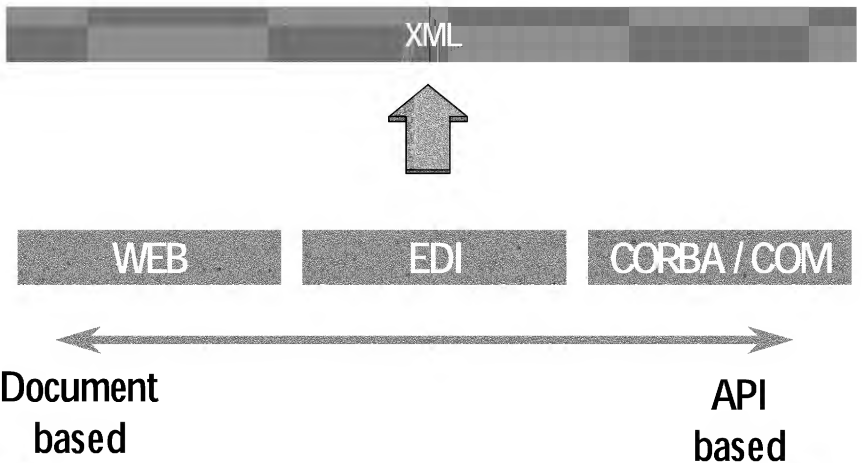
The XML Revolution

- Today's Web sites publish information for people
  - "eyeballs-only" is dominant design perspective
  - hard to search
  - hard to automate processing
    (too much "scraping and hoping")
- Tomorrow's sites will provide information and services for computers (and people)
  - overcomes HTML's inherent limitations
  - enables the new business models of the network economy

XML as Technology Platform

...exchange data in an application and vendor neutral format

...the simplicity of HTML with the precision of APIs



Commerce Networks ⟷ Shared Information Models

- Supply Chains
  - Merchants, distributors, manufacturers, brokers, logistics, shippers
- Real Estate
  - Brokers, banks, escrow, title, inspection, MLS, government agencies, classifieds, loan aggregators
- Securities
  - Brokers, financial advisors, markets, research services, account management
- Travel
  - Hotels, airlines, rental car agencies, travel agents

Laptop Description Seen "By Eye"

Laptop Computer

IBM Thinkpad 560X

233 Mhz

32 Mb

4 Gb

4.1 pounds

$3200

HTML Laptop Description

```
<TITLE>Laptop Computer</TITLE>
<BODY>
<UL>
<LI>IBM Thinkpad 560X
<LI>233 Mhz
 <LI>32 Mb
 <LI>4 Gb
 <LI>4.1 pounds
 <LI>$3200
</UL></BODY>
```

XML Laptop Description

```
<COMPUTER TYPE="LAPTOP">
<MANUFACTURER>IBM</MANUFACTURER>
<LINE>Thinkpad</LINE>
<MODEL>560X</MODEL>
<SPEED UNIT="MHZ">233</SPEED>
 <MEMORY UNIT="MB">32</MEMORY>
 <DISK UNIT="GB">4</DISK>
 <WEIGHT UNIT="POUND">4.1 </WEIGHT>
 <PRICE CURRENCY="USD">3200</PRICE>
</COMPUTER>
```

Smarter Processing Enabled by XML

- Shared schema for laptops, desktops, and towers
- <COMPUTER> provides a logical container for extracted and manipulating product information as a unit
  - Sort by <MANUFACTURER>, <SPEED>, <WEIGHT>, <PRICE>
- Explicit identification of each part enables its automated processing
  - Convert <PRICE> from "USD" to French Francs, Italian Lira, etc.

Airline Schedule Seen "By Eye"

Airline Schedule

Flight Information

United Airlines #200

San Francisco

11:30

Honolulu

2:30

$368.50

HTML Airline Schedule

```
<Title>Airline Schedule</Title>
<Body>
<H2>Flight Information</H2>
<H3>United Airlines #200</H3>
<UL><LI>San Francisco
<LI>11:30
<LI>Honolulu
<LI>2:30
<LI>$368.50
</UL></Body>
```

Airline Schedule in XML

```
<TransportSchedule  Type="Airline">
<Segment Id="United Airlines #200">
 <Origin>San Francisco</Origin>
<DepartTime TZ="PST">11:30 </DepartTime>
 <Destination>Honolulu</Destination>
<ArriveTime TZ="HST"> 2:30 </ArriveTime>
<Price Currency="USD">368.50</Price>
</Segment>
</TransportSchedule>
```

Example: Schema for Transport

Using the same schema for all scheduled transportation services:

<TransportSchedule Type="Airline">
<TransportSchedule Type="Train">
<TransportSchedule Type="Ferry">

An application could create itineraries that involve more than one service by matching on locations and times

Shared Semantics for Time and Location

Shared semantics for location and time in all schemas that need them enables richer "commerce networks" of services:

<TransportSchedule Type="**Airline**"> ...
<Destination>**Honolulu**</Destination>

<Accommodation Type="**Hotel**">...
<Destination>**Honolulu**</Destination>

<Event Type="**Concert**">...
<Destination>**Honolulu**</Destination>

Domain-Specific Commerce Languages

R-665

Domain-Specific Languages for Commerce Networks

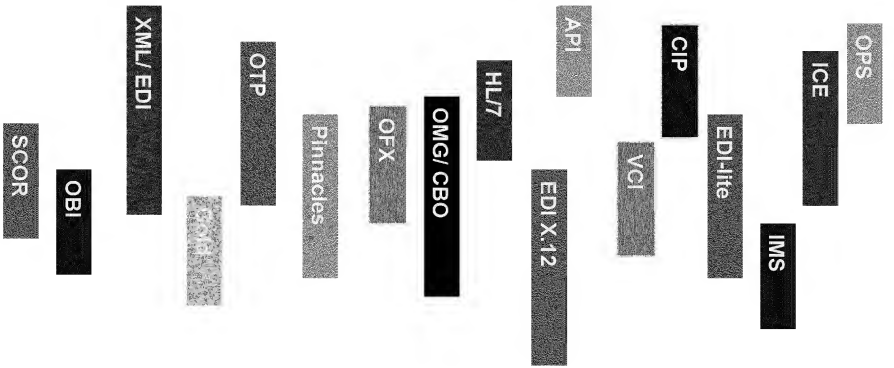
OBI	Corporate Procurement	AMEX, Office Depot, Boise Cascade
OTP	Retail Payment	Mastercard, Mondex
OFX / GOLD	Personal Finance	(Intuit, Microsoft), (IBM, 125 Banks)
ECOM	Computer Supply Chain	Ingram + 24 largest channel players
ICE	Content syndication	News Corp., Sun, Microsoft, Adobe, Vignette, C/Net

This list is growing explosively, and all are using XML (or shortly will be)...

XML and Metcalfe's Law

- XML makes it easy to create markup languages
- But the value of a language depends on how many people (or computers) understand it
- How do you encourage and enable others to understand your language?
- The EDI approach:
  - BIG COMPANY: Speak MY language or I won't do business with you!
  - SMALL COMPANY: Yes, master.
- The XML approach:
  - Excuse me, here are the rules of my language if you'd like to speak with me…

Tower of Babel - Stovepipe Protocols



- Delayed time to market
- Redundant development costs
- Limited Interoperability

The Common Business Library

R-669

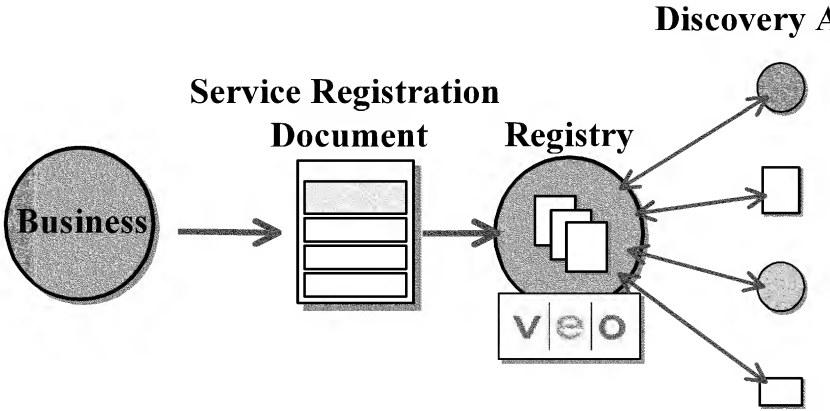
"Loose Coupling" via Shared Document Definitions

- Interconnect business systems and services in terms of the documents they exchange rather than in terms of their application interfaces

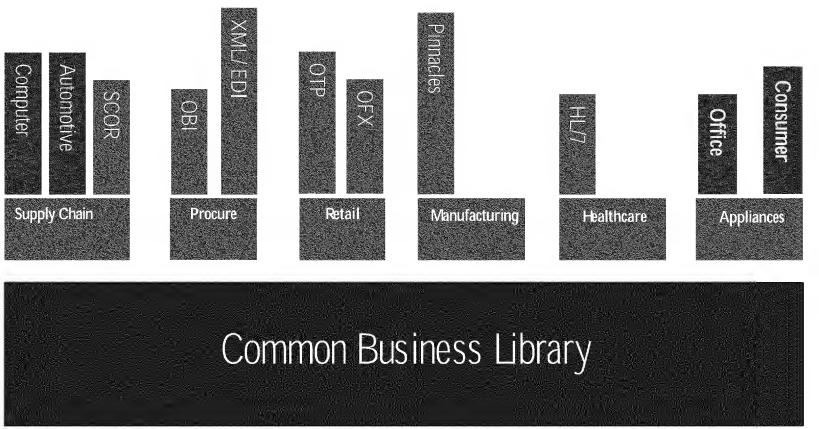
- Shared document definitions provide an intuitive framework for specifying the business logic and computations that take place on each end of the exchange.

- Five shared document definitions are implied in these two business rules:
  - if you send me a **request** for a catalog, I will send you a **catalog**
  - if you send me a **purchase order** and I can fulfill it, I will send you a **shipping notice** and an **invoice**

Businesses Publish Services Using XML Documents



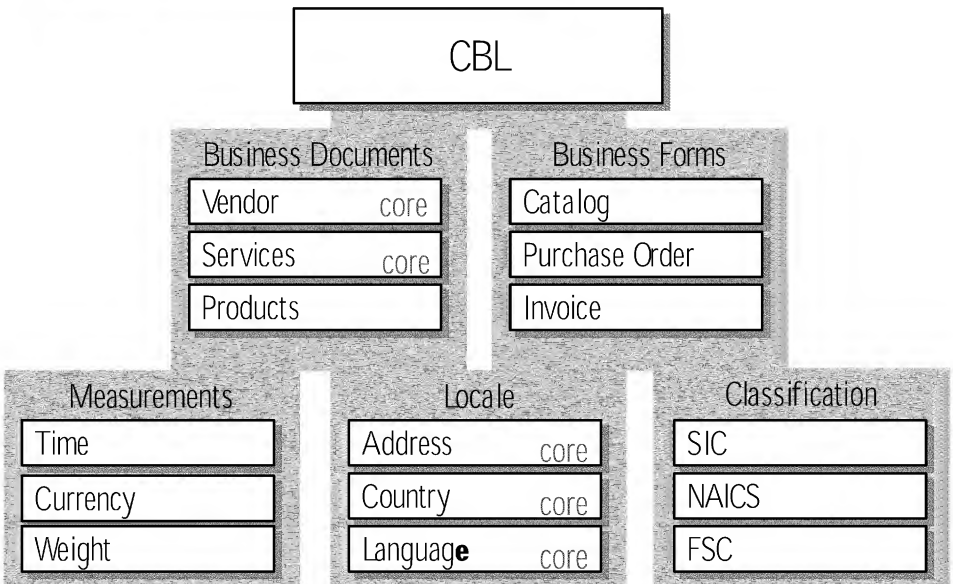
Open Framework For Commerce



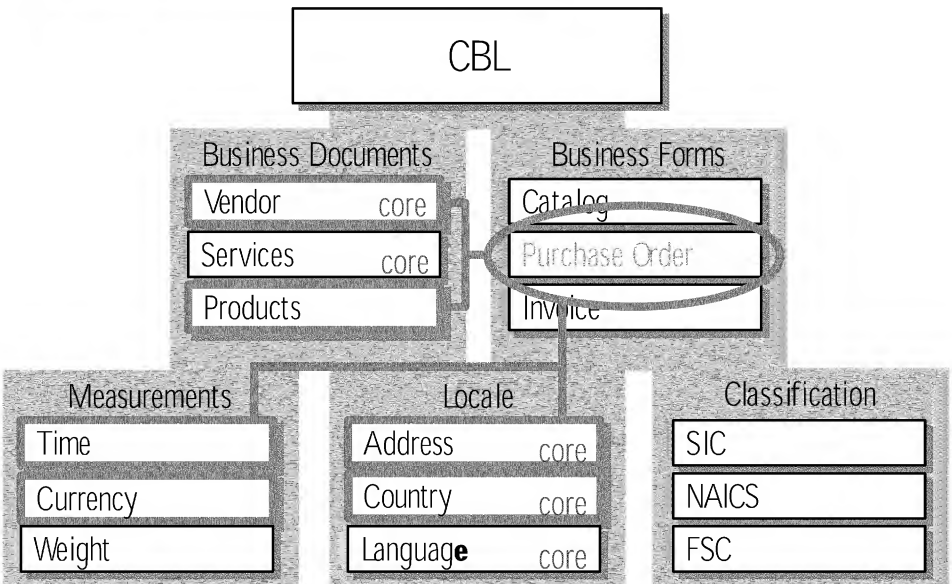
The Common Business Library

- The functions and information that are common to all business domains, building on existing standards or conventions
- Specifies common semantics, common syntax, and message packaging
- CBL documents are described by XML DTDs to make them "self-descriptive" and validatable
- Complex descriptions and messages can be composed from primitives
- Domain-specific XML applications can be implemented in "native" form or as "hybrids" for maximal interoperability

Building Blocks



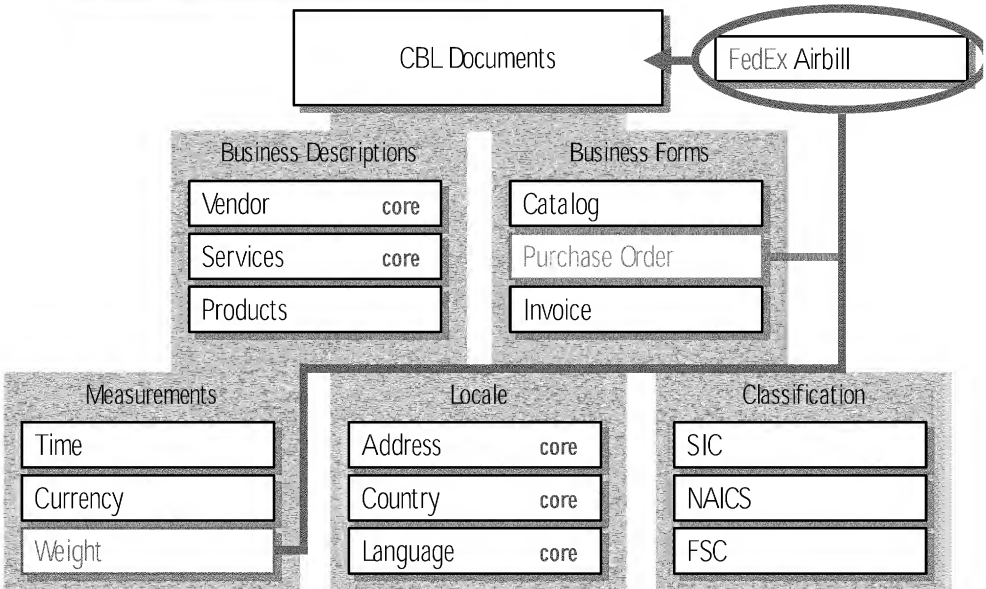
Building Blocks



Document Type Definitions and Modules

- addresso.mod, for Geographical Information
- catentry.dtd, for a Simple Catalogue Entry
- contact.mod, for Contact Information
- countrys.mod, for Country Codes
- datetime.mod, for Date and Time markpart.dtd, for Market Participant Information
- markdesc.dtd, for a Market Description
- pay.mod, for Payment Information
- proddesc.dtd, for a Simple Product Description
- shipment.mod, for Shipping Information
- shopcart.dtd, for a Shopping Cart
- transact.dtd, for Transaction Documents

CBL Building Blocks



Business Services Described Using CBL

```
<service>
<service.name>Order Service</service.name>
<service.location>www.veosystems.com/order</service.location>
<service.op>
        <service.op.name>Submit Order</service.op.name>
        <service.op.inputdoc>po.dtd</service.op.inputdoc>
        <service.op.outputdoc>poack.dtd</service.op.outputdoc>
</service.op>
< service.op>
        < service.op.name>Track Order</service.op.name>
        <service.op.inputdoc>request.track.dtd<service.op.inputdoc>
        <service.op.outputdoc>response.track.dtd<service.op.outputdoc>
</service.op>
</service>
```

CBL Status

- CBL v1.0 contains a few dozen DTDs and modules developed from analysis of ISO, ANSI X.12, other standards
- CBL currently being used by Veo Systems in demonstration applications (Project Seitai, GSA catalog interoperability)
- CBL to be starting "fodder" for CommerceNet-sponsored WG to develop open framework for interoperability of domain-specific commerce languages (just getting under way)

The Economist

"Untangling the Web"

25 April 1998

.."But the biggest role that XML is expected to play is in integrating the way that existing paper documents -- invoices, loan applications, contracts, insurance claims, you name it are exchanged between organizations around the world. Imagine what the world would be like if one company's computer system could automatically read any other organization's documents - and make complete sense of them? This is the goal that the technique known as EDI has struggled, unsuccessfully, to achieve for years. Though efforts have barely begun, there is a chance that XML could actually make that happen. If it did, business on the Web could run riot."

Exhibit J. Excerpts from W3C "note" WSDL version 1.1 (March 15, 2001) accessed at http://www.w3.org/TR/wsdl

W3C

Web Services Description Language (WSDL) 1.1

W3C Note 15 March 2001

This version:
http://www.w3.org/TR/2001/NOTE-wsdl-20010315
Latest version:
http://www.w3.org/TR/wsdl
Authors (alphabetically):
Erik Christensen, Microsoft
Francisco Curbera, IBM Research
Greg Meredith, Microsoft
Sanjiva Weerawarana, IBM Research

Abstract

WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services). WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate, however, the only bindings described in this document describe how to use WSDL in conjunction with SOAP 1.1, HTTP GET/POST, and MIME.

Status

This document is a submission to the World Wide Web Consortium (see Submission Request, W3C Staff Comment) as a suggestion for describing services for the W3C XML Activity on XML Protocols. For a full list of all acknowledged Submissions, please see Acknowledged Submissions to W3C.

This draft represents the current thinking with regard to descriptions of services within Ariba, IBM and Microsoft. It consolidates concepts found in NASSL, SCL, and SDL (earlier proposals in this space).

This document is a NOTE made available by the W3C for discussion only. Publication of this Note by W3C indicates no endorsement by W3C or the W3C Team, or any W3C Members. W3C has had no editorial control over the preparation of this Note. This document is a work in progress and may be updated, replaced, or rendered obsolete by other documents at any time.

A list of current W3C technical documents can be found at the Technical Reports page.

Table of Contents

1 Introduction.
1.1 WSDL Document Example
1.2 Notational Conventions
2 Service Definition
2.1 Document Structure
2.1.1 Document Naming and Linking
2.1.2 Authoring Style
2.1.3 Language Extensibility and Binding
2.1.4 Documentation
2.2 Types
2.3 Messages
2.3.1 Message Parts
2.3.2 Abstract vs. Concrete Messages
2.4 Port Types
2.4.1 One-way Operation

R-682

2.4.2 Request-response Operation.
2.4.3 Solicit-response Operation
2.4.4 Notification Operation
2.4.5 Names of Elements within an Operation
2.4.6 Parameter Order within an Operation
2.5 Bindings
2.6 Ports
2.7 Services
3 SOAP Binding
3.1 SOAP Examples
3.2 How the SOAP Binding Extends WSDL
3.3 soap:binding
3.4 soap:operation
3.5 soap:body
3.6 soap:fault
3.7 soap:header and soap:headerfault
3.8 soap:address
4 HTTP GET & POST Binding
4.1 HTTP GET/POST Examples
4.2 How the HTTP GET/POST Binding Extends WSDL
4.3 http:address
4.4 http:binding
4.5 http:operation
4.6 http:urlEncoded
4.7 http:urlReplacement
5 MIME Binding
5.1 MIME Binding example
5.2 How the MIME Binding extends WSDL
5.3 mime:content
5.4 mime:multipartRelated
5.5 soap:body
5.6 mime:mimeXml
6 References
A.1 Notes on URIs
A.1.1 XML namespaces & schema locations
A.1.2 Relative URIs
A.1.3 Generating URIs
A.2 Wire format for WSDL examples
A.2.1 Example 1
A.3 Location of Extensibility Elements
A.4 Schemas
A.4.1 WSDL Schema
A.4.2 SOAP Binding Schema
A.4.3 HTTP Binding Schema
A.4.4 MIME Binding Schema

1. Introduction

As communications protocols and message formats are standardized in the web community, it becomes increasingly possible and important to be able to describe the communications in some structured way. WSDL addresses this need by defining an XML grammar for describing network services as collections of communication endpoints capable of exchanging messages. WSDL service definitions provide documentation for distributed systems and serve as a recipe for automating the details involved in applications communication.

A WSDL document defines **services** as collections of network endpoints, or **ports**. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. This allows the reuse of abstract definitions: **messages**, which are abstract descriptions of the data being exchanged, and **port types** which are abstract collections of **operations**. The concrete protocol and data format specifications for a particular port type constitutes a reusable **binding**. A port is defined by associating a network address with a reusable binding, and a collection of ports define a service. Hence, a WSDL document uses the following elements in the definition of network services:

- **Types**– a container for data type definitions using some type system (such as XSD).
- **Message**– an abstract, typed definition of the data being communicated.
- **Operation**– an abstract description of an action supported by the service.
- **Port Type**–an abstract set of operations supported by one or more endpoints.
- **Binding**– a concrete protocol and data format specification for a particular port type.
- **Port**– a single endpoint defined as a combination of a binding and a network address.
- **Service**– a collection of related endpoints.

These elements are described in detail in Section 2. It is important to observe that WSDL does not introduce a new type definition language. WSDL recognizes the need for rich type systems for describing message formats, and supports the XML Schemas specification (XSD) [11] as its canonical type system. However, since it is unreasonable to expect a single type system grammar to be used to describe all message formats present and future, WSDL allows using other type definition languages via extensibility.

In addition, WSDL defines a common **binding** mechanism. This is used to attach a specific protocol or data format or structure to an abstract message, operation, or endpoint. It allows the reuse of abstract definitions.

In addition to the core service definition framework, this specification introduces specific **binding extensions** for the following protocols and message formats:

- SOAP 1.1 (see Section 3)
- HTTP GET / POST (see Section 4)
- MIME (see Section 5)

Although defined within this document, the above language extensions are layered on top of the core service definition framework. Nothing precludes the use of other binding extensions with WSDL.

1.2 WSDL Document Example

The following example shows the WSDL definition of a simple service providing stock quotes. The service supports a single operation called GetLastTradePrice, which is deployed using the SOAP 1.1 protocol over HTTP. The request takes a ticker symbol of type string, and returns the price as a float. A detailed description of the elements used in this definition can be found in Section 2 (core language) and Section 3 (SOAP binding).

This example uses a fixed XML format instead of the SOAP encoding (for an example using the SOAP encoding, see Example 4).

Example 1 SOAP 1.1 Request/Response via HTTP

```
<?xml version="1.0"?>
<definitions name="StockQuote"

targetNamespace="http://example.com/stockquote.wsdl"
          xmlns:tns="http://example.com/stockquote.wsdl"
          xmlns:xsd1="http://example.com/stockquote.xsd"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
          xmlns="http://schemas.xmlsoap.org/wsdl/">

    <types>
       <schema targetNamespace="http://example.com/stockquote.xsd"
              xmlns="http://www.w3.org/2000/10/XMLSchema">
          <element name="tradePriceRequest">
             <complexType>
                <all>
                   <element name="tickerSymbol" type="string"/>
                </all>
             </complexType>
          </element>
          <element name="TradePrice">
             <complexType>
                <all>
                   <element name="price" type="float"/>
                </all>
             </complexType>
          </element>
       </schema>
    </types>

    <message name="GetLastTradePriceInput">
        <part name="body" element="xsd1:TradePriceRequest"/>
    </message>

    <message name="GetLastTradePriceOutput">
        <part name="body" element="xsd1:TradePrice"/>
    </message>

    <portType name="StockQuotePortType">
        <operation name="GetLastTradePrice">
           <input message="tns:GetLastTradePriceInput"/>
           <output message="tns:GetLastTradePriceOutput"/>
        </operation>
    </portType>
```

```
    <binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="GetLastTradePrice">
           <soap:operation soapAction="http://example.com/GetLastTradePrice"/>
           <input>
              <soap:body use="literal"/>
           </input>
           <output>
              <soap:body use="literal"/>
           </output>
        </operation>
    </binding>

    <service name="StockQuoteService">
        <documentation>My first service</documentation>
        <port name="StockQuotePort" binding="tns:StockQuoteBinding">
           <soap:address location="http://example.com/stockquote"/>
        </port>
    </service>

</definitions>
```

1.2 Notational Conventions

1. The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [2].

2. The following namespace prefixes are used throughout this document:

prefix	namespace URI	definition
wsdl	http://schemas.xmlsoap.org/wsdl/	WSDL namespace for WSDL framework.
soap	http://schemas.xmlsoap.org/wsdl/soap/	WSDL namespace for WSDL SOAP binding.
http	http://schemas.xmlsoap.org/wsdl/http/	WSDL namespace for WSDL HTTP GET & POST binding.
mime	http://schemas.xmlsoap.org/wsdl/mime/	WSDL namespace for WSDL MIME binding.
soapenc	http://schemas.xmlsoap.org/soap/encoding/	Encoding namespace as defined by SOAP 1.1 [8].
soapenv	http://schemas.xmlsoap.org/soap/envelope/	Envelope namespace as defined by SOAP 1.1 [8].
xsi	http://www.w3.org/2000/10/XMLSchema-instance	Instance namespace as defined by XSD [10].
xsd	http://www.w3.org/2000/10/XMLSchema	Schema namespace as defined by XSD [10].
tns	(various)	The "this namespace" (tns) prefix is used as a convention to refer to the current document.
(other)	(various)	All other namespace prefixes are samples only. In particular, URIs starting with "http://example.com" represent some application-dependent or context-dependent URI [4].

3. This specification uses an **informal syntax** to describe the XML grammar of a WSDL document:

- The syntax appears as an XML instance, but the values indicate the data types instead of values.
- Characters are appended to elements and attributes as follows: "?" (0 or 1), "*" (0 or more), "+" (1 or more).
- Elements names ending in "..." (such as <element.../> or <element...>) indicate that elements/attributes irrelevant to the context are being omitted.
- Grammar in bold has not been introduced earlier in the document, or is of particular interest in an example.
- <-- extensibility element --> is a placeholder for elements from some "other" namespace (like ##other in XSD).

R-685

Exhibit K. Glushko, LaPlante et al., Document Standards & Technologies for Commerce Applications (delivered Sept. 1, 1998) accessed at http://www.google.com/search?q=cache:hq5pefHRwksJ:seminars.seyboldreports.com/1998_san_francisco/ETAPE_26.html+%22cbl+1.0%22+veo&hl=en&gl=us&ct=clnk&cd=7 on Oct. 26, 2006, 28 pp.

Seybold Seminars Online

Seybold San Francisco/Publishing '98 Web Publishing Conference

Document Standards & Technologies for Commerce Applications

Tuesday, September 1, 1998

Mary LaPlante, Fastwater, LLP, Moderator
Jaime Ellertson, Interleaf, Inc.
Robert Glushko, **VEO** Systems
Eric Severson, IBM Global Services
Audience Q&A

Mary LaPlante: Welcome to day of the corporate Internet-extranet track here at Seybold. My name is Mary LaPlante. I am one of the partners in Fastwater, a research and consulting firm that looks at the way the Web is affecting companies' business processes, models and relationships. It's my pleasure to serve as the chair of the corporate track and I'm also your moderator this morning for this first session on the second day of the program. At the start of the track yesterday morning, we highlighted three key themes that our speakers are addressing and discussing in the sessions in the track. First, business processes and the immersion of publishing activities directly into them. Second, electronic commerce and the powerful connection between content and commerce and what it makes possible for our organizations. And third, relationships with customers, partners and competitors and how corporate publishing is adapting to facilitate and to drive those relationships. These themes are central to the

development of the emerging network economy and they reflect the transition that corporate publishing is undergoing as companies adapt to new challenges and opportunities. We continue to examine these themes and the tracks that comprise today's program as well. We'll start this morning with a session on electronic commerce and the document standards and technologies that are related to building commerce applications in your organizations. After lunch we'll have a session on publishing on extranet and what's involved in reaching outside of your walls but still to a private network, and we'll wrap up today with a session that looks at the integration of customer data and publishing processes; not publishing output, which we've been talking around, but actually publishing processes--how do you get documents through your organization? So our first session this morning is actually about one of my very favorite topics: money. Specifically, it's about conducting commerce and transactions using Internet technologies inside of our organizations. The goal of this session is to help you make connections between document systems and commerce applications. It's kind of a nuts-and-bolts sessions, but I don't mean nuts and bolts to refer to a technical session. What I mean is this is where you're going to learn how to really make some of this stuff work. How do commerce applications and document systems technologies and standards play together? Where's the added value that they bring? What kinds of unique capabilities are afforded by a particular document system, a particular standard? How are they being cost-justified by the companies who have attempted to deploy them? We have three terrific speakers, who are going to explore these issues from three very different perspectives. Jaime Ellertson is president of Interleaf, and I have asked Jaime to examine the issue of the role that content plays in commerce application and how it can be leveraged. Bob Glushko is director of component-based commerce for **VEO** Systems, and I have asked Bob to look at the issue of interoperability of systems and processes, what kinds of languages to we need to facilitate communication and commerce. And Eric [Severson], who is executive consultant at IBM's global services, is going to address the key strategic issues involved with wedding document systems and eommerce applications. So at this point, I am going to welcome our first speaker to the podium. Please join me in welcoming Jaime Ellertson. [Applause]

Jaime Ellertson: Thank you, Mary. As Mary said, my name is Jaime Ellertson. I am going to speak on XML-based document standards and technologies for commerce applieations. My one advert in this whole day is just to give you a little bit of background on myself and Interleaf, more importantly, Interleaf. It's a public company, a large software company that focuses on complex publishing and XML-based content management solutions. And in the discussion today, you'll hear about examples of companies, but these are typical eustomers for us--everything from manufacturers who are dealing with content and reuse, high-mix, low-volume goods like airplanes, where the information is critical and lengthy, to financial institutions. pharmaceuticals. I am going to try to start today with a little bit of

humor and a business problem to drive home what content can do today over the Internet, so my set-up for this is not exactly a current day. The year is 2019, I'm Jaime Houston, a manager for Androids Are Us, which is a distributor for the Bi-Rail Corporation. I don't know how many of you remember the movie Blade Runner, but this is a slight take-off on that. And Bi-Rail Corporation's theme is androids are more human than human. In other words, each one is different and the documentation and information, service, repair and warranty information is pretty extensive on that many mechanical parts. I utilized an XML-based content management system provided by my android manufacturer, Bi-Rail Corporation, over the Web to do a lot of my service ordering, supplier information, and Bi-Rail's system allows me to perform many of the basic functions that function internally today, but of course the Web is a standard is 2019. You'll notice I haven't moved past Windows '95. But the Web is the standard for connecting and communicating, both with my customers and with my suppliers. So with no further adieu:

"I want to ship 200,000 units by this time tomorrow. I am saying Houston, we have a problem. Let me bring you up to speed. About an hour ago, one of the new Nexus 6 androids went off in shipping. It seems that for no reason at all, this Nexus 6 named Leroy started to malfunction. First he started by moving a little slower, which seemed to affect his target points, which left him a couple of feet shy of the shelving area. Then he would race around and hyperjive like some sort of possessed Energizer bunny. Finally, he started tripping, and I mean tripping. The unit was finally subdued by our creative warehouse staff, who administered a little attitude adjustment. So what do you think?"

Well, it appears I have a problem. I have got a customer with a very expensive product that I supply and it's malfunctioning. In this year, 2019, I ean quickly sign on to Bi-Rail's content management-based Web site, get the service section, sign on with my password and user ID, and behind the scenes Bi-Rail Corporation, with XML-based content management system, creates and analyzes my user ID and builds a list of the products that I have purchased. And it immediately tells me that the items I have purchased include a Roy batty, which happens to be for this warehouse agent. In this case, I can immediately move to the service manual, go ahead and select the service manual, and go into a little bit down within the service manual that's just been created to the quick trouble-shooting guide. And here I'll go ahead and identify that the robot is stumbling, that the rotating mechanism needs to be replaced therefore, and move straight to the reference. I can walk that customer through real time all of this information being created dynamically because every robot is different. The rotating-foot mechanism, what they're going to need to do to replace it, and then move down to an online ordering system. Now again we're in the year 2019 and we haven't moved past the symbol of shopping carts, but in this case, I go ahead and select that

product. It automatically fills out the order form for me. My manager of shipping is John Pavlov. I fill in a PO number and off I go. It tells me my order's been processed, and in the year 2019, that product will arrive in three hours by virtual sled. So I'm in good shape and I think I've probably satisfied my customer's problem.

"As usual, you came through for me, steward. Thanks."

So let me go over the background of that, XML-based content management. Quite simply, in the old-world, document management was a blob--a bigger entity, limited reuse, generic content beeause it was all molded together. In a content level scenario, personalized content, best-of-breed elements, your graphic artist versus engineering, and it quite often is going to come in the form of everything from headers, footers, illustrations, graphs, to different components. Likewise it's going to come from different sources. Your engineer may create a better CAD/CAM graphic of the rotating foot mechanism, your tech doc person or a marketing person may put together a better product description. In terms of an overview, the original source documents end up being broken into their components, not stored as an entire blob. They are indexed and stored in some form of repository and then by using a style language and the DTDs, you can go ahead and apply them for the different output purposes.

The Web, as some of us refer to it as a very long roll of toilet paper with links between it, is a different format output than an 8 1/2x11 sheet or a CD-ROM. So a different style can be applied to the same data but for a different purpose. In terms of XML and where that comes in, XML is an open neutral standard from the W3C. It is more than HTML in that HTML only really describes the layout where they're tied, and XML deseribes the meaning and the structure, the content and the context. So as we go through it, tagging something in XML would mean that you can actually identify the content elements by type: a header, a footer, a rock star--Prince versus Prince Island Sound or Prince Edward's.

And XML tagging supports intelligent use of the information. The DTD allows me to, as I build that information and input it, ensure that the structure and the content of the document for the most part is what my DTD, my document type definition, describes. In this case, I didn't fill out the fund family, it requires it, so if I move forward and fill that out, then I have a document that matches the document type definition and the structure of the other documents that I'm going to compare this to, so I have an ability to compare apples to apples.

In terms of indexing this information typically in a content management system, then I'm going to pass the information, those elements into a repository, and those repositories can then store and tag and manage that information both on a content basis, so I can

search for a word, or contextually. I can search for all of the headers that have size 14 or that talk about a rock star because my tag says rock star titles, and then I know when I'm searching in that field for Prince that it is a rock star, a musician. And in terms of the output, I can apply XSL or a style language for the specific outputs.

As I've already talked about, the Web is different from print. In terms of my example, in a complex manufacturer, financial institution, or service bureau, you can see that allowing your content to be reused once you've built that, especially in the Androids-Are-Us example, is beneficial because I need it to build the actual product, I need that information for my suppliers to share it with my distributors, and I need for my customers. So the content management can bind those three together in what we'll call a value chain.

So the one question that constantly comes up that we hear when we're talking to these large, early adopters in this area with XML is how do I get my content in? You're not really expecting me to sit down and write my documents in XML. And the truth is we don't, we don't see that. This typical process I've already talked about is a structured document tag, repository, and then using styling, but how do you get it into a structured format? How do you get in XML? Are you going really input an XML-based product on everyone's desktop? Probably not. We see the markets emerging.

One protyping tool that we have out today allows you to take a standard document type definition and generate a word template so that the generic user in your organization is using the most common UI available, which happens to be Word today on about 75 million desktops.

How does that look? Well, let's take the example of Bi-Rail Corporation again, and I'm going to pretend that I'm a supplier now and I'm the person supplying those robotic foot elements. In this case, I would download the parts template. It would look like this. Notice that's a Word document. At the top left, I have save as XML, validate part document, preview the XML, and then e-mail the XML into that. So that as a supplier to Bi-Rail Corporation, I can input the document with my related product that I'm supplying to them and they can get it in a structured format where apples are going to be compared to apples.

Recently, with the CIOs at Lockheed and for an average F22 fighter jet, they have 68,000 suppliers. I would call that a paper-information nightmare in terms of getting that information in and then supplying it back for service, repair, warranty, priee lists, whatever. In this case, that's a standard template. It's a Word template. Here's what it looks like filled out. If I want to go ahead and validate that then, I can go ahead and submit it for validation. It comes back and tells me that I've forgotten the vendor name, and as you can see up in the vendor box,

I've forgotten to fill out Robocorp, which is who I am. So I go ahead and fill in Robocorp, I go ahead and submit it back, and now the validation is successful.

Now I have valid XML, not well-formed XML, but valid XML that applies to a DTD, and when another suppliers submits their parts, it's going to be the same information that they compare apples to apples for, and when they know they want a price book, and every product has to have a description, it will produce that. In this case, if I ever wanted to see the XML, I could go ahead and see it. It's not necessarily pretty, but there's valid XML for that exact document that you're looking at generated from a Word template. And then, if I wanted to, I could file that or e-mail it back to the company to submit with my product.

So the business benefits, that process of getting a content management system up to support your information on a Web site and communicate with customers, distributors, that whole supply and distribution chain. It's pretty substantial when it gets to XML, and this is where you're very Microsoft, and that's David [Doby], all of the players getting behind it because XML data not only is intelligent but it will allow you to do things like mass customization for a market of one, protecting your value chain, binding the distributor and document enabling eommeree, because the documents can aetually compute that information or ask the server to compute it and only send down intelligent data.

I can ask the server only for the jeans size 34 at that vendor Web site and he'll send me down only that list beeause it's able to cull the data because it's already tagged. and with the metadata, it can make that analysis and present it for me. In this ease, mass customization, you know, I can serve a market of a million on my Web site, but each individual that signs on. like myself, it will check the password, the profile, and then present the appropriate data. It also can allow the customer to act upon that information and interact with the data.

Financial community? A simple example is I can publish from the same database a hundred different reports, depending on your high-net-worth portfolio. In terms of value for manufacturers, they see a value chain as being tying the information back to them. If I'm Toyota, I find that it's amazing, but Toyota is able to sell me a car for $25,000 in the showroom, but if I go buy it in spare parts from the spare-part window, it would cost me about a million five. Somewhere in there there's some profit in those spare parts for those guys and they want you to come back to them for years after they've even finished on your warranty, so binding you with a content management system that allows a manufacturer to buy that airplane spare part of anything else through the manufacturer, not going around to NAP auto parts to buy it, is very important. And that's what we talk about.

And when we see a corrupted value chain, we see a situation where the distributor or even the end customer just decides to go willy-nilly to a second party. The content management, as you saw with the Androids Are Us example, if it's that easy for me to trouble shoot the repair, order the part and get it in three hours, I'm going to buy it from Androids Are Us. I'm not going to go through a second-tier shopper for a second-party part on each stick.

So the content management system combined the distributor and the original customer, and the benefits are pretty clear. It binds the customer and the distributor to the manufacturer, it integrates that value chain, simplifies transactions and information sharing. It can effectively block competitors out and leverages current IT investments. Obviously, in that example, I went ahead and filed my HTML order form back with Bi-Rail Corporation and it connected through it SAP system, filled out the order, shipped the product and confirmed with a confirmation back to me. And as we said and showed you previously, it even knew what products I had purchased because the data was intelligent. Thank you very much. [Applause]

Ms. LaPlante: In the description for this session, we talk about a couple of different types of systems and technologies that are required to work together to actually do commerce. There's content management, obviously. There are customer data management systems, there are your business legacy systems, there are information systems, transaction processing. In your experience, Jaime, when you're talking to your customers who are out there thinking about making these things all work together, where do they start? Where are they thinking about starting in terms of building the connections between the systems that sort of need to work together?

Mr. Ellertson: Well, I think in a number of the customers, and I'll use manufactures as an example, Mary, they have to start at two places. One, a company like Lockheed, which is a leading defense and aerospace company, obviously. Actually, when I went down to see them recently, they had their number one initiative as E-Commerce, and I kind of shook my head and said you're not really going to sell spare parts for 300-million-dollar jets over the Web, are you? But their focus was just on filling the supply chaining because that's where their most significant problem is in communications and linking up their external suppliers is a bigger problem for them. So they analyzed the situation from a problem standpoint and what would smooth their ability to do greater volumes of revenue. And then we have seen in the financial community the delivery points be the focus. With all that said, they all seem to have an issue with getting the information up and the volume of information that they have to convert into either a document management, a Web site, or a content management based system, and so the other comment I'd make generally is that I see issues with identifying a set of structures

or that metadata, the DTDs, and then building those to build the content management or Web related system.

Ms. LaPlante: Go ahead.

Robert Glushko: One of the nice things about dealing with a company the size of Lockheed is that they have lots of things going on. We've been talking to Lockheed too and they tell us that their most important E-Commerce problem is getting the parts of Lockheed to do business with each other. Lockheed has grown over the last several years by buying lots of other companies and there's about 50 of them now that call themselves Lockheed as an umbrella, and each of those different business has different computing environments and processes and they're looking at using XML in this kind of model that you heard about as a way to interconnect those different business units. So things as simple as time-keeping and reporting and budgeting and so on are really pretty difficult when you've got the pieces coming out of different business units that use different technologies, so they're using XML as a way to think about how to expose common documents from the different business units even though they may be produced by different business processes and technologies.

Ms. LaPlante: Anyone in our audience have a question? Questions for Jaime? Yes?

Audience: [Inaudible]

Mr. Ellertson: Yes. That product with a better presentation than I give with a real android is downstairs in the exhibit hall. It's called Blade Runner and you can see it.

Ms. LaPlante: Other questions? I'm sorry.

Mr. Ellertson: I'm sorry?

Audience: [Inaudible]

Mr. Ellertson: Yes. One of the bigger things, in fact Lockheed was a good example, and I think you'll hear about it at this show when people talk about XML, is the idea of getting XML in. If everyone has to sit down and write XML, it's going to be a long time before we get a lot of XML on the Web, and I have got an article from PC World here that talks about that as being one of the significant issues. So at Loekheed, for instance, we spent time with them and they said we'd like to think Interleaf is a supplier of a very high-end publishing system that we could buy 85,000 copies of Interleaf, but they said guess again. What we have on the desktop is a standard desktop. It includes a browser, Word, the Office Suite and some other tools, and we have to have a mechanism that allows the engineer, the general

marketing guy, the guy in purchasing to all input into the system and get structured information so we can compare that and then dynamically build the Web information and share it across the corporation, and if it's a neutral standard like XML, that's great. And so we focused an approach on Word being the input vehicle.

Ms. LaPlante: Let me see if there's another question from the audience because then I'm going to turn the podium over to you, okay? Anybody else? Yes.

Audience: [Audience]

Ms. LaPlante: Could you use a microphone in the back there, please, so everybody could hear you?

Audience: I have a Web site with about 65,000 documents and I was wondering what would it take for me convert that site over to this system.

Mr. Ellertson: Well, I'm going to try to stay away from an advertisement for my product specifically, but there are products out there, and Mary, myself, probably a lot of the panelists could tell you about that that indeed do conversions. But converting back-end data, people have products out today that will, especially if it's more structured data, and convert it either into SGML, HTML, or XML now, and they're essentially a conversion process, so that's one way of doing it. This approaeh happened to focus on going forward with Net new documents.

Mr. Glushko: I just have to chime in and make two points in response to Jaime's last point in that question, which is that actually there's a tremendous amount of latent XML already out there because many of the Web sites that are out there are exposing as HTML information that's coming from a database or other structured information source, which could be XML if we had a way to actually render it. So we're really close to being able to have a lot of XML on the Web that right now is being dummified into HTML by our CGI scripts and things like that. And related to that is the last question, it's very conceivable that of those 65,000 documents, many of them in fact are coming out of structured sources, so rather than trying to fix the current manifestations of those documents in their HTML form, you want to look back to where they're eoming from. They may be coming from databases and it's much more efficient and robust to go baek to those sources and transform those sources to XML than trying to go through the HTML you have in the middle. It's like giving yourself a lobotomy and then trying to get smart again. You're better off just going back to the original sourcc and trying to project that into XML.

Ms. LaPlante: I think we'll invite Bob up now, but please join me in

thanking Jaime for his presentation first. [Applause] Our second speaker is Bob Glushko from **VEO** Systems.

Mr. Glushko: Thank you very much, Mary. I want to apologize up front for my slides, which seem to have had the fonts shrink on them in the last 24 hours, when I asked my assistant to make sure the fonts were sort of the portable variety, so they went from being 32 or 36 point to being 24 point, and if it's portability, then we have a problem. The title of my talk is "Implementing Domain-Specific Commerce Languages with a Common Business Library," and the problem we're trying to solve is how to get all those little XML applications we're inventing to get along, so it's sort of the Rodney King talk.

The basic idea is that we're seeing XML emerge as the technology platform for commerce applications and a lot of companies are investing in developing what we call domain-specific languages for commerce. By that we mean they're developing their set of DTDs and modules that describe the kinds of messages they want to use for commerce, things like catalogs and requests for inventory and payment, and purchase orders and other kinds of documents of commerce. And we've been working on a strategy to make all of those documents get along better by building them from the same basic XML components.

I work for a company called **VEO** Systems, which is a for-profit spin-off of the Commerce Net Consortium. We were originally called CN Group. We were created partly in response to an opportunity to have your tax dollars fund this product that I'm going to tell you about. There's a part of the Department of Commerce called the Advanced Technology Program which gives money to start-up companies to try to do basic research that, if it's successful, will have a great commercial pay-off, and we proposed a strategy of using XML as the infrastructure for interoperable commerce applications and they gave us some of your money to do that, and I thank you very much for that. We've changed our name recently to **VEO** Systems. Those of you who know Spanish or Latin, **veo** means I see, I understand, because that's a good name for an interoperable commerce company, and we are growing real fast, and if you're a capable developer in XML and Java, I have a job for you in Silicon Valley.

The basic idea of the XML revolution--I really think it is a revolution. Very few things have taken off as fast as XML has. And the revolution is that we are moving from a Web for eyeballs, in which designing for eyeballs is and dominant design perspective, which of course leads to the problems that things are hard to search and hard to automate because HTML just isn't very good at that, to a view of the Web that is going to be for just eyeballs and computers. We'll use XML to create an infrastructure that machines can understand as well as eyeballs can view, and that I think is the key for new kinds of business models of the network economy, to use the phrase that Mary

used. The network economy is all about more efficient ways of building virtual companies, where companies can share business services with less friction. You can bolt a shipping service to a virtual catalog and have a virtual company even though you have no company at all, for example. So in our view, the basic idea of XML as a technology platform is that we have a market-oriented technology, so it has the kind of mass appeal for sixth graders and grandmothers of HTML that they can do mark-up. At the same time, it has the precision of more application or API-based coupling, so you can use XML to express a very precise interface through some kind of service. It's kind of document-based computing where the documents define the business services that we want to provide, and I'll explain more about that in a bit.

But first let me review a little bit about why XML is so critical here if you haven't got a firm grasp of it. This is the computer that I'm using to display this presentation. I really paid $3,200 for it. This morning it's on sale for $1,900, and it's only been nine months later. That's a bit of a problem, but that's another issue. So if I want to search for this computer on the Web, I could go to lots of places and try to search for it. The problem is that searching is going to be done by fairly brittle technology that does what I might call scraping and hoping, that's a scrape and hope because what it really finds on the Web is something like this. What it can get access to at most sites is an HTML rendition of this underlying structure that has things like list items in it, and so I can't really say I want to find a computer that weighs less than 3 kilograms, costs less than two million yen, and that has at least 500 megabytes of storage, which this one does, but I can't find that because I can't match that unusual description if it's in a way which is marked up in HTML. However, if I would use XML and mark it up with a document model like this one, I could do all sorts of great things with it. Here I've invented an XML application that I call computer, which is a computer mark-up language. And I have tags like manufacturer and product line and model and speed and things like that in which I've encoded not just the values but the units of the values, so I could in fact do all sorts of great things like have from a computer store download a candidate set of computers that might match my basic query and then sort them on the basis of manufacturer or price over the speed or whatever. I could convert the price to whatever unit I'm interested in. If I'm coming from Japan, I could turn it into yen. I could convert the units from kilograms to pounds, and so on. So I can have a client-side processing technique because I have smart data. I haven't given it a lobotomy to turn it into HTML. Likewise, if I want to go off to a vacation to Hawaii, I may want to find airline flights, but if I'm looking at a Web site that has HTML, I see something like this, which is a flight on United Airlines between San Francisco and Honolulu, I can't even tell from there which direction it's flying. My eyeballs can make some guesses, but under the hood, there's nothing that tells me about which is the origin and what's the destination. What I want is something like this: XML data marked up in a schema like transportation schedule, where I have

explicitly tagged information about origin and destination and departure time and arrival time and so on because with that kind of information, I could do useful things. I could search, I could bring the entire United Airlines schedule to my browser and do all of the optimizations on my client without having to thrash back and forth waiting for Effort to talk to Expedia or Travelocity or United Airlines, or whatever it is. But what's more is that I cleverly choose the document scheme that I use to encode this information. In this case, I've used transportation schedule, which is more general than just airlines. Any transportation schedule service could use this DTD, so I could say let me use trains and ferries and BART and subways and any kind of other scheduled services, and I could have an automated process find me the lowest cost or the fastest schedule, or whatever kinds of itinerary I'm trying to optimize all programmatically because I can have shared semantics for place and time and money and stuff like that. And if I were to be even more clever and find a way to mark up my data not just about transportation in this way but let's say hotels or events and have them use equivalent tags for describing place and time, you can see how I could have an automated program on the Web that could find me the cheapest ticket to Hawaii over spring break, a hotel for three days after I landed, and buy me a ticket to any concert that has an artist top 40 all automatically, because I could line up the doeuments, in a sense, on the basis of the overlapping tags.

And that's what the network economy is all about, in my view. We're building commeree networks, kind of trading eommunities, on the basis of shared information lots. So what is a supply chain but a buneh of companies that have agreed to talk about the same product in the same way? You know, real communities are basically brokers, banks, escrow companies, title companies, inspcctions, multiple listing serviees, and so on that have agreed to talk about the same house in the same way and the same buyer in the same way and the same loan package in the same way. So we've got to find ways to define those common document models for different communities of commerce. And that's what domain-specific languages are all about, so we're seeing a lot of activity here. Some of you have even heard of some of these alphabet soup consortia likc OBI or OTP or OFX or ECOM or Ice or Rosetta. These are all groups of eompanies have gotten together to develop specifications of the messages and documents that they need to do business in their particular communities. So, for example, the Rosetta Net specs or the ECOM specs is a set of specifications being developed by companies in the electronic industry computer supply chain. Ingram Micro is organizing this effort to get companies like Hewlett Packard and Compaq and IBM to talk the same way about their products so you ean have more efficient communication between the manufacturers and the stores that sell their products, and all of the other things in between: inventory, prieing, promotion and so on. This is a great idea but it's also a terrible idea. You see, XML makes easy-to-create mark-up languages. That's the good thing. The bad thing is that it makes it

easy to create mark-up languages. And the value of a mark-up language or any language in general depends on the number of people that speak it, so obviously, languages like English, that a lot of people speak in the world, are more useful than languages like Danish, that is spoken by about three million people. Nothing against Denmark. I used to be married to a Dane, but there's not many people who speak Danish. So we need to find ways to get companies to speak the same commerce languages. But we already have a way. It's called EDI. Big companies pounded their fists on the table and said you'll speak my language or we won't do business. Well, that's a little too coercive and it makes it hard to invest in speaking more than a few languages if it's expensive to learn them, which EDI is. On the other hand, XML is a little bit too polite. With XML, a company says here's my commerce language, I've invented a mark-up language for my products. Does anyone eare to talk to me? If we're not lucky, the whole world will invent its commerce language and we'll have the same problem of the Tower of Babel. We'll have no languages that are mutually intelligible. So we're looking at a future in which we have these various stove-pipe protocols for OBI and XML, EDI and HL7 and OTP and Ice and you name it, and they're all here at the conference talking about these great protocols and they won't work together, so we'll all invent them ourselves, we'll have a delayed time to market, redundant effort, and we won't get along.

So the goal of our common business library is to solve this problem, and it's built around a view of business that says businesses do business by exchanging documents. That's a looser way to do business than to connect to your system through your APIs, but it's better to couple loosely than not at all. The basie idea is that companies will tell each other what documents they need to do business. They'll say if you send me a request for a catalog, I'll send you my catalog. If you send me a purchase order, I'll send you an invoice. And we express those document definitions in terms of XML, which is publicly exposed, like on a Web site saying here's my DTDs that I want to produce and consume. And the key of that is the common business library, in which we've taken those various stove-pipe protocols like XML, EDI, OBI, OTP and XYZ and whatever else and looked at what they have in common. Well, they have a lot in common. They all talk about place and time and measurement and currency and so on. So we've built as a starting point something called the common business library, which is about a thousand different elements organized in several dozen DTDs that have taken things like international standards for measurement, currency, time and so on and encoded them in a certain style of XML. And from these building blocks, we can define more complicated documents for commerce. So, for example, most of the things you would use in EDI, in ANSI X-12, if you know what that means, can be modeled by building up from pieces of this library. So, for example, the purchase order is basically a document that consists of fragments that describe the vendor, its products, and has some descriptions of when these things have to go where. Likewise, if I'm a new company that wants

to play in this kind of deal, I can say, oh, I know, I'll expose my business services in terms of a document interface, so--I'll skip this. Those are specific examples of the kinds of fragments that are out there. You can see that we have things like an address mod for describing geographical information, a catalog module for simple catalog, currency and so on. These are little building blocks that you can use in your applications. So a Federal Express air bill might be modeled as a purchase order that has extra information about weight. And now anyone can send a request to Federal Express with an XML document by conforming to that new DTD, which they have made publicly available.

So our vision of how this works is as follows: Your company will publish on its Web site essentially a little corner that is the XML corner that will say here's how I want to do business electronically with you. I want to have a service, I'll call it the ordering service, that has two transactions. The first one is a submit transaction, the second one is a transaction, and basically, I'm saying if you send me a document that conforms to po.dtd, a purchase order, I will send you back a purchase order aeknowledgement. And these DTDs could be defined in some global registry or in some commerce registry or some other places all over the world--they should be freely available--that simply say how you want to do business automatically. Well, this is CBL. We have basically just released it to the world this week and after having used it in several demonstration projects, one with the federal government and one with the consortium of Japanese companies, and we have decided to find a way to make this most publicly available and most robust as possible. So we've started a group along with Commerce Net. We're calling it the Echo Framework Group because it derives from something called the Echo Vision, the vision of plug-and-play component commerce that we started a couple of years ago, and the idea is that we've invited technical people from the major commerce corporations, the IBMs, the Suns, the Netscapes, the Microsofts and so on, and from the major commerce specifications like the OBIs and the OTPs and the Ices and so on, and we're going to hash out a CBL that is more broadly based than just our year or two of work, and everyone will have access to this stuff. So we'll all be able to build XML applications for commerce out of the same sort of semantic DNA, so that our documents will be mutually intelligible. This effort is actually being announced at this conference. I'm sort of giving you a sneak preview one day early, but you came to this dark room and you get something for that. And if you want to see a sneak preview of CBL, give me a business card. It will be up on our Web site in about four or five days, probably the day after Labor Day, but I'm not sure because we're moving and you know what that means. But the basie idea is that we're going to have **CBL 1.0** available to the masses, you, and we're giving it to this working group, which will then try to take it and generalize it and ensure that OTP and OBI and Ice and everybody else can use it to describe their applications.

Now we actually have a technology business wrapped around this, which I won't say very much about except that the idea is that if you have common business building blocks, you can define you own document models, expose them to the world, and then have a process which takes those things in and out of your legacy systems. So you see where it says the **VEO** processor, the bump on the right in green. Documents come in from the outside world through those service definitions and then get parsed and routed through things like catalogs and ERP systems, E-Commerce systems of various kinds, and third-party services get worked on and then they come back and get sent back as XML documents. So if you're sending a purchase order, all I'm telling you is if you give me a purchase order and I can handle it, I'll send you back an invoice. But let me go on. Inside of my company it may be pretty complicated. I may have to send it to my order-entry system, look you up in my customer database, talk to my bank if I want to give you credit, check my inventory to see if I actually have those products.

[Turn Tape]

And all of our processing gets done invisible to you because all it explores is the document interface, and then I send you back the document that I promised to send you back, namely my invoice and my shipping schedule. This vision seems so simple that even the Economist business magazine figured it out. They had an article a couple of weeks ago called "Untangling the Web, All About XML," in which the reporter said 'you know, if companies could extend documents around and they would be understood, we'd actually have an EDI that worked, and XML may be in fact the way to do that.' I actually have a vision which goes farther than that. I think that the fax machine, which is the simplest possible way to couple your business to the world, should spit out an XML document so that you can couple to the world by just plugging in your fax machine and it says I'm here to do business and I will send you the following kinds of documents when I do business, and that will really change the world for electronic commerce, that kind of easy, low-cost coupling where you exchange documents which are mutually intelligible. And I think something like the common business library is a key part of that and I encourage you to get on that initiative. Thanks. [Applause]

Ms. LaPlante: So a common question that I always get when we talk about XML from users trying to figure out what to do with it is okay, so I make my own language, my business partners make their own language, and how are they going to work together? So common business language is meant, as a starting point, to sort of lower the playing field, and the Echo Working Group is going to be where all of this expertise is going to come together?

Mr. Glushko: Right. In some sense, the benefit of XML by itself is

that it enables you to say explicitly what your documents are about. So you're saying my documents follow this DTD. Now if two companies have different DTDs, they've got to figure out what the relationship is. You may call it pounds and I may call it kilos; we've got to find a way to make those things talk about weight in the same way. But because we're explicit about it, we can do a mapping function which says oh, multiply mine by 2.2 and we're done. So being explicit is a good thing. But if everyone starts from a common place, we don't have to do that mapping. So our goal with the common business library is to make it possible for people to start at reasonable places. I mean, common business library has things like the 168 currency codes from the Ice organization. I mean, USD means United States dollars, FFR means Freneh francs. Sure you can invent your own language for talking about that, but why would you bother? We're simply doing kind of a syntactic unification of the world standards for commerce in a way that can be used by XML applications.

Ms. LaPlante: Any questions for Bob?

Mr. Glushko: Like I said, anyone who wants to can just give me a business card and I'll send you mail when it's up on my Web site, or just come to www.veosystems.com in a week or two and it probably is going to be there, but I'd rather have your business card so I can give you better treatment.

Ms. LaPlante: Okay, I wanted to go back to something that Jaime said, where you were shaking your head violently a moment ago. You both agree that a content management is important, essential to making all of this work. You've got to have the stuff structured and stored in some way. You both agree that XML is obviously the way to do this. Jaime happens to believe that by strengthening the distribution chain, the value chain, you're going to block competition. Bob, you shook your head there because your model is if we're all using XML and you make known how to do business with you, competition is going to be freer, broader, wilder. Talk to me.

Mr. Glushko: Okay. Well, one problem with talking about electronic commerce and open markets and having explicit DTDs for describe your products and services is that if you have a consumer model of that, you're thinking that's bad, they can find my price more easily and compare me with everybody else on the same kind of stuff. If we all use XML to describe our books, then they can find the lowest price supplier faster with XML, and that may be bad. First of all, what XML really gives you is the chance to provide richer functionality on your Web site, so you can have, for example, an applet on your Web site that would convert your prices to some other currency, whieh would be a good thing even if you were explicit, so you might find that more beneficial than the cost of being value compared. But fundamentally, this is about business-to-business commerce. In five

years, the business-to-business communication using XML will be 50 times larger than that using XML for consumer applications because it's really plumbing business-to-business connections. And those may be private trading networks that are using it, but we're not using EDI. It's just that we're going to reduce the cost of getting in. Right now the couplings cost of EDI even for a small player could be 10s of thousands of dollars. Our vision is that--I think my Web site, whatever happens, I could have my tiny Web site now with a little corner of it that's XML-ified, and now I can play with the automated gang, and it's taken me an afternoon to create a couple of DTDs and map some of my basic product descriptions to an XML format. So we're reducing the cost of coupling in trading communities and I think that's a benefit which outweighs the possibility of being commoditized in some way.

Mr. Ellertson: Mary, I don't want to create agreement when there's healthy dissent, but I think in reality, I'm in vehement agreement with your first statement, which is strengthening the value chain as we talk about it doesn't necessarily block competition because XML blocks it, it blocks competition because I as a customer of that android can get my service information twice as fast because my XML site is dynamic, because when I sign on, instead of getting some 4000-page manual that tells me everything about all of the androids, I get my specific manual. If I have a Toyota Camry, when I sign on Toyota's Web site, I don't want to spend 50 minutes finding the Toyota Camry model 670 in that four models, I want to get them to produce my Web site. When I sign on Fidelity's site, I want to get my statement, especially today, and find out how much money I've lost with the markets moving. And so I think there is substantial agreement there. If there's any maybe healthy disagreement, it's that we focus that the issue with XML right now isn't that XML has value making your data structured and intelligent is going to substantially move E-Commerce forward, it's how are you going to get all of this data into an XML format. And I happen to concur that a common business language would be great, but there are some issues of getting everyone to try to agree. Other people have tried to do that. Microsoft seems to do it with a little stronger arm than other people or capability because of the desktop, so we happen to be focusing on that problem of getting information converted into valid, not just well-formed because if it doesn't conform to a DTD, you've got, as we just heard, one person's description of their E-Commerce document, purchase order, and the next company's example, so it's getting some standards, which some of these groups are setting and other people are going after, and then getting all of that data into XML so it's all intelligent and we're all plugging apples to apples, and we think in the past, the SGML issues and history have created a big logjam, so we're focusing on that issue of getting information converted into XML freely, openly through a standard UI.

Mr. Glushko: And one other comment that I would make. I liked

your picture with the corrupted value chain very much because I think part of the picture here is what is possible to do with the technology. You have to have a technology there if you're going to do any of these interesting things. The second part of the picture is can we agree on how we're describing business objects and so forth, or else the technology doesn't do us any good. But the third part of the picture is now that we have all of this at our disposal, what is the intelligent thing to do from a business standpoint, and we can see examples on the Web right now of business models kind of gone awry and run amok, where people are not making money in the way that they thought they were going to make or they're giving away free information when that wasn't really the idea, etc., and I think we have to put that whole pieture together with this idea of am I enhancing or corrupting my value chain.

Ms. LaPlante: We have one question in the back, and then we'll--

Audience: Is XML itself stable politically and otherwise?

Mr. Glushko: I sort of think XML is stable enough. There's a proposed recommendation from the W3C that came out in February that people are doing. I think there's always--you could always take a specification and try to aggressively implement or we could say let's use the sort of core stuff, which is not likely to change, and what we're trying to solve is agreement on basic tag sets to simplify the concept, and our library uses primitive concepts and actually doesn't push the envelope in terms of using XML as a specification. We're doing things that you could do on your Web site today with very common, easy tools. We're not pushing the envelope with XML. XML is undergoing a lot of changes. There's a major effort coming down the road which is an XML schema language, which would let you describe more data typing and semantic information inside of the XML definition, and that will be really important for commerce. But I'm confident that players in the W3C and the XML industry in general have their heads screwed on right here and they recognize that the proprietary gains are in the past. For example, our little working group we just started has Microsoft, Sun and Netscape in it because they recognize that XML will fail unless they all get along at some basic level. They may have private browser wars, but fundamentally, they're saying this is a good idea.

Ms. LaPlante: Let's hold additional questions and discussions until Eric has a chance to deliver his presentation because I think what he has to say is going to add to this conversation. So please join me in welcoming Eric [ieverson]. [Applause]

Eric Severson: Thank you very much, Mary. As Mary said at the beginning, I'm going to talk about some of the more strategic issues kind of from a consultant and system integrator's point of view about how this technology should be used intelligently. What are the true

business issues that relate to document standards in this area? I come from a group in IBM, global services, that is basically the center of competency for electronic or enterprise document management, electronic publishing, XML, SGML, kind of the intersection of those sorts of things, and we do full life cycle consulting and systems integration so tend to see things from soup to nuts, and I'm going to talk in a moment about a particular project that we have just completed that I think is a good case study for looking at this area. The philosopher J.B.S. Haldane said at one point a number of years ago, 'the universe is not only stranger than we imagine, it is actually stranger than we are even capable of imagining.' The Saturday Night Life philosopher, Roseanne Roseannadana, put it this way: 'Jane, I'm telling yah, it's always something.' And I think that there's a huge aspect of the Web and all of this stuff that we're looking at today that has that kind of disconnective kind of character, and so we call it a revolution. On the other hand, I think that there's a way to look at this as sort of a more natural evolution of how we've used computers in automating business issues starting from the administrative kind of systems and going through office automation and connectivity. Eventually, the office automation, which had a lot to do with documents, and the connectivity that was e-mail and so forth, were going to result in something like documents on the Web. But the problem that I think we have strategically is that the Web culture and the revolution that is the Web doesn't quite jibe with what business is looking to do. And I always like to quote a Forbes article from what now is several years ago, but they talked about the genius of Mark Andreeson, who of course was the programmer behind Mosaic and then the chief technical officer of Netscape, who was credited with seeing when other computer visionaries like Bill Joy of Sun and Ted Nelson, the father of hypertext and so forth, when they said yes, we can see an Internet and a Web but it's not ready for prime time, you can't guarantee that the data would have integrity, you can't guarantee that the links will have integrity, you can't guarantee performance, and the genius of Mr. Andreeson was that he saw that it didn't matter, that people would stay up all night if that's what it took to get the performance, that people would follow, as they say, they'll gravitate to the half of the links that aren't broken. And they even drew this romantic kind of cowboy image of interviewing Mark Andreeson down highway 101 with half-eaten cheeseburgers strewn across his back seat just like the half-broken links on the Web. well, he was right, by gosh, and the Web succeeded, but when we're talking about within the next couple of years that nearly all mission-critical applications are going to be on the Web for business, that doesn't quite fit. The business data is not quite the same as all of the excitement of interconnectivity that the general Web offers to us. Business data, especially that which is mission-critical, absolutely has to be correct. It's not just that you can get to it, you've got to be able to trust it. Oftentimes the documents are long-lived. They go through complex change cycles that are hard to manage, and without managing them very carefully, you don't know that what you're looking at on the Web is correct. It's not just the Web often that's

being supported, it's other kinds of formats, it's interactions with other kinds of business systems within the organization, and so the bottom line for this sort of E-Business vision is that no matter how cool the Web sites may be, no matter how exciting some of these ideas are, we've got to be able to rely on the data.

And so the first technology that I want to make a strong point about is the document management work flow technologies and other kinds of things that have to do with managing the change cycle for creation, approval, and update of information are absolutely essential to be able to come up with a vision on the surface that most of us would see of the Web, of having information that we can actually trust, and so this kind of architecture of document management and control source data kind of behind the scenes is absolutely crucial. It is something that for the most part Web masters and the Web culture do not fully appreciate yet, but in this idea of Web content management is starting to really come alive now, and it's certainly very consistent with what the other speakers have said this morning.

Now what about XML? XML offers us new opportunities not only to control this source data but to start really leveraging what we can get out of the business data. And there are a number of different things, only some of which I have actually listed here before I ran out of room, but one of the things certainly that has been a key part of XML has been this idea that I can describe my HTML documents using metadata or attributes that then allow me to do more intelligent searching across the Web. It allows me to push content in a personalized way by matching the attributes of the document with the user profiles of the consumer of that information in ways have also been suggested earlier in this session. XML is a language that can give kind of a universal interface between documents or Web pages and databases, which are the heart of business information. Hooks for document management and work-flow processes and so forth. But the last one is the one I want to focus on today with a real example, which is the idea of delivering intelligent data to the Web client or to the consumer of that information. As Bob quoted, I think, not just for eyeballs and brains to process, but data that computers can process and link up to other systems. John Bosak, who's the chair of the W3C committee, has talked about the various advantages of XML, and one of the things that he's pointed out is that with the tremendous bandwidth issues and performance issues on the Web that XML, as one of its good points, can offload a bunch of that processing to the client side. As he puts it, XML give Java something to do. There's enough raw material there that a lot of client processing can be done, it doesn't always have to go back to the server. And I think that's an extremely important kind of technical point about the Web infrastructure, but I'd like to take that one step further by saying it isn't just that you can do client processing instead of having to do it on the server, there is a type of client processing that is extremely important that could only be done on the client beeause that's where

the data live. I want to talk about this idea of taking this kind of public data that's on the Web and linking it up with the private data that is actually the province of the information consumer.

And to do that I want to take a real project that we've just completed with Mazda Motor Corporation, first tell you where that is at right now, and then talk about some of the possible extensions that XML would afford. We haven't yet implemented XML with Mazda, but there are a number of interesting possibilities where we could go from where we're at. As just a way of quick background, this is a service bulletin and other kind of service application out to the Mazda dealerships, and not surprisingly, Mazda's customer satisfaction depends heavily on how well cars can be serviced and whether they need to be serviced, but certainly if they do, how fast people can be back on the road at a minimum cost. And that depends on dealerships really knowing what they're doing, and to facilitate that, Mazda publishes service bulletins, as all the automotive manufacturers do, and a number of different kinds of service information to help those dealers. But up until now, this has been totally paper-based. It's a classic loose-leaf publishing kind of application where updates come and people are supposed to put things into three-ring binders, throw away the old one, and so forth, but in order to really understand what's in there, not only do they have to do all those updates, but they have to realize that there was something in there that they need to access, and if they don't even realize it's there, they won't find it. So embarking on this project, Mazda set out to replace all of that paper and that loose-leaf update kind of process with a Web interfaee where service bulletins could simply be looked up on the Web. And this was not only a matter of lowering costs and getting rid of paper, but mostly a matter of really improving overall customer service through the dealership channel. And what it looks like now, it's been piloted and is now getting fully rolled out and deployed, is basically the same kind of idea, where from most of the consumers of the information point of view, this is a Web browser interface, but they're able to rely on the fact that this data is correct and get to the right stuff because the search attributes and the document management kind of controls and so forth are operating behind the scenes to make sure that this really works. So this is great stuff, and we've issued several press releases about this, and Mazda is extremely happy with the result.

But now imagine with me how we might extend this using some of these ideas about XML we've been talking about this morning. One very simple example and just one possibility here that I want to just go through is an idea of an E-Commerce example with linking up the public service bulletin information, public in the sense at least that it's across all of the Mazda dealers, to the private data that the dealer has about its own inventory. With the current HTML interface, people are able to now get on the Web browser and look at things like a brake replacement procedure and see, for example, that they need to replace part XYZ, but the first question the mechanic is going to have is

okay, so do we have part XYZ? And because this is, as Bob put it, only for eyeballs, basically one could look at an inventory report, one could go say, hey, Joe or Shirley or whatever, do you know, do we have something, could you go find it, and so forth, but there's no real way to know when you're looking right at that procedure whether you have an XYZ or not, and there's no way that Mazda could publish that or serve up a Java applet or anything else that would tell you that because Mazda doesn't know either. It's only the dealership itself that would know. With XML, it would be possible to actually tag that part number as a part number, so one of the sort of common business library or language kind of ideas might be that we all agree on what's a part number, at least among first of all the Mazda dealerships. And by having that little tag in there rather than just simply the text saying replace part XYZ, it is suddenly possible to automatically have a transaction that takes that and queries the local inventory control system to see what the inventory status is and pop right up on the screen for that mechanic the status of that part. Furthermore, if it happens to be out of stock, business rules could be executed that are unique to that dealership that say in that case, I have a network of distributors where I might be able to get a part shipped to me overnight, and I'm going to initiate automatically over the Web, if the user wishes me to, an XML-based transaction now back again to public data or maybe the private data of that distributor network that tells me where is the nearest place I could get a part shipped from. And if I decide to go ahead, in this case it says that there's one in Phoenix, then I could initiate an XML-based order right back to that server and have a part shipped overnight, thereby completing an E-Commerce transaction. And one can look at this and start to see an evolution where you could say well, gosh, if that's true, maybe even people in a related industry could trade this kind of information with each other, and maybe it's not just the distributors I know about, maybe it's any distributor that's posted the kind of transaction Bob was talking about, for example, on the Web, and I could go query across the Web and look for anybody who has basically set themselves up as somebody who could be a purveyor of this part. And so I think that the possibilities are endless even from this one example, and it all depended on one tiny little tag that simply said this is a part number rather than just displaying it for someone who can look at it and say oh, yes, I know that's a part number.

So in summary, and again I want to emphasize this is just one potential that I just wanted to go through to give you a concrete example of how this really works and why HTML by itself can't do it, but XML allows this application-specific information to be part of the data that's actually being delivered over the Web, so you can see it but you can also compute with it and link it to other applications that the server and the publisher of this data has no access to. It's a matter of providing the raw material to link these business applications together, and linking the public data with private data to provide a fully linked E-Commerce kind of transaction network. It's information sharing, it's information retrieval, it's the way towards

personalized service by matching up attributes of the consumer and perhaps even their data as expressed through an XML interface with what is out there on the Web that could be served up to them, and really the means for full integration of E-Commerce or E-Business.

And lastly, I just want to say, as I like to say, if we're so smart, than how come we aren't out there already? Why is this just an extension that's possible, and just point out that, you know, the question was asked of the standards mature. I personally believe that XML itself is stable politically and otherwise, but there are some other related standards that are not quite there yet. Certainly, the tools and the products are starting to appear or coming online quickly but they're not quite there, and I think that will be changing very quickly over the next year or so. The industry standard XML applications or the common business library kinds of ideas are just starting to be put in place, but I think we'll see rapid progress of that. And finally, I just want to emphasize most of all, and this is a viewpoint that I really have gotten as a consultant and an integrator that I didn't quite have when I was in a previous life as a software vendor, and that's just how hard it is to make any progress from a system standpoint, how hard it is for people to change, how hard it is for processes to change, and that you have to take one step at a time. And again, just to re-emphasize the other point I was making a moment ago, you also want to take one step at a time because in a sense, with this incredible explosion of technology, we're playing with fire, and one does want to have an enhancement of value chain, not a corrupted value chain, and it makes some sense to kind of go a phase at a time and expand this one step at a time. Thank you very much. [Applause]

Ms. LaPlante: Eric, one of the things that's real unique about XML and makes it real powerful is the fact that it can be used in two real different ways. One is a protocol language to actually conduct business processes, and the second is a document encoding standard, a way to put tags around information. In the consulting and the interactions you have with your customers and prospects, do they have their heads around that concept that XML is more than just some kind of newfangled SGML, that it's got this whole other processing side to it?

Mr. Severson: Well, I think people are looking at XML from different angles depending on their background, their interests and so forth, and there isn't a common understanding of what XML is. those of us that come from an SGML and structured documentation background certainly view XML as a simplification of [XML] that makes it much more usable, puts less in the way of rapid success, and certainly that is true. I think those who come not from that background at all but simply from a Web background are looking at XML also in a couple of ways, depending on their particular focus. One way is XML as a more dynamic, flexible and extensible use of HTML. In itself, it's not HTML plus-plus, etc., but it provides that

kind of a function for the Web. But there are other ways of looking at it that are equally interesting. One way, for example, is to say from an EDI perspective over the Web, XML is a pure ASCII format that can be validated from top to bottom, can't contain viruses, can't contain other kinds of Trojan horses that could violate security and fire-law concerns just by the way it's set up. Therefore, it is a much, much better way to do E-Commerce on that basis than something like a JDBC interface, which is all binary encoded, and so forth. So there are many views, and I see a bunch of those at once going on.

Ms. LaPlante: In the conversations you and I had preparing for this session, you keep reminding me that linking back to business objectives is what you do when you work with your clients. What kind of acceptance at the executive level, the people you interaet with on a regular basis, what kind of acceptance of XML are you finding? I mean, you and I have both been in the business of trying to get SGML adopted and we always ended up talking to the tech pubs manager. What about XML? Are exeeutives getting the message?

Mr. Severson: Well, I think that like the Web itself, there are kind of two levels of how executives look at the Web or XML. There's the very rational level, which is really what I keep reminding you of that I try to help my clients think through, which is what are we trying to do here for our business? I don't care if any of this technology existed, where are we trying to go, why are we trying to do it, and what are the really key points and the simplest possible way of looking at changing the business to get there? Maybe a revolutionary change to the business is justified, but what's the most straightforward way to look at it? Now how does technology like XML or the Web fit into that? How can we map that together? That's the rational view, and I think that executives will accept that. Sometimes it takes a bit to go through that, but the other more irrational or emotional kind of view is XML is all over the press, it looks like it solves everything, why aren't we using it? And that's kind of cool for those who are promoters of XML to get that kind of attention. On the other hand, it's kind of dangerous because there's always the chance for a backlash if it doesn't meet the expectations, and I think a lot of what certainly I try to do and in general I think the industry needs to do is kind of balance that to this is just a technology. It's a very powerful technology, it has a lot of potential uses, but it's just a technology. Let's keep the focus on what are businesses are really needing to do.

Mr. Glushko: I want to say one thing there about my experience with explaining XML to senior executives in corporations. I think that HTML is the best thing that ever happened to us and for the reason that Eric said, that it was broken but it became ubiquitous because it was so simple. And we both paid our dues as SGML people. In fact, all of us have paid our dues as SGML people. We have a lot of blood on our hands from trying to do SGML for the last decade, and it was hard because the idea of marked-up and document centered

computing was a new idea. Now that executive in a corporation either has played with the Web him or herself or has a kid that does it, and the idea of mark-up as a way to express some kind of document or interface is pretty straightforward. In fact, XML is more straightforward than HTML. When you say you have a recipe, how do you want to describe it, you think in terms of elements for ingredients and measurement and stuff, and it's a lot easier to think that way than saying how do I turn it into LIs and ULs. So that I'm finding it easy to explain XML to business people because you get the benefits of that hard-core computing infrastructure like Cobra and stuff, and yet you can look at it and make sense of it like a Web page. So it's this wonderful kind of duality. Now again, a lot of magic is hidden under the hood there, but the idea of a document interface is a lot more like what you do when you do business. It's like can we agree on our contract? It's a whole lot more sensible for a business person to understand than can we make our APIs interoperate. I mean, that's geek talk, right?

Ms. LaPlante: We have some questions in the back. Yes?

Audience: I'm Dan Applequist from theStreet.com. We're developing a publishing system right now or the idea of a publishing system based on XML, and we're doing some DTD development as part of that process, and I guess my question is two-fold, one of which is what resources are out there to assist in DTD development, and another one is in trying to find the resources myself that are available for DTD development, especially for documents in XML, I keep running into the SGML academia wall. You know, as soon as you scratch the surface, you're looking at DTDs that were developed to support scientific publishing, and those aren't addressing the business issues that we have. So what efforts are going on in that arena?

Mr. Severson: I think that there are certainly resources out there to do DTD development. We do that in our group, for example, but I think you're right that there's--let me explain quickly what causes the SGML academia kind of phenomenon that you're referring to other than individual personalities and so forth. The issue that people who have been in the SGML world are worried about is that you go through all of this effort to encode your data and all of this conversion like we were talking about--that's the biggest issue out there--and you do it in such a simplistic way that just a couple of years or less down the pike you realize that you've got to do it over again. And so SGML was based on this idea that model all of the various things that are in the data because you never know when you might need one of those as a critical hook. But that has led to almost an obsession with data modeling that doesn't quite get back to a common sense, rational approach of well, yes, but we can't model it forever and we've got to get some real results, and let's not make this too complicated either, so there's a balancing point that I think has been missing. I actually had an argument with one of my colleagues where they said well, but

if I don't do this, how do I know that their data will be okay in 10 years. And I said can you name one other thing you think you can predict in 10 years other than their data, and if you can't, then I've got a question whether you actually can predict anything about that. And we sort of worked it back. I said what about five years, what about three years, and at two years started to feel pretty confident. No, no, no, I think I know what's going to happen here. So I said well, then I think what I've learned from this is if you're trying to think ahead more than about two or three years, you're probably barking up the wrong tree anyway. So I guess what I would just say is there are resources out there, but it is extremely important to be in control of the sort of balance about your own application rather than let somebody tell you you've got to model it down to every nth degree.

Mr. Glushko: Well this is precisely the problem that we set out to solve with the common business library. We've got the world's best SGML designer developing this stuff, okay, but what we're trying to do is not do it as a classical big, mother-of-all-document models for here's the commerce model. We're saying we're not sending around huge 50,000-page tech manuals for Boeing, we're sending around lots of little commerce pieces like purchase orders and catalog fragments and invoices and stuff. Those are much simpler documents. There's a whole lot of them, but there's a much simple model set of them, and so it lent itself more to this kind of componentized SGML-XML modeling exercise. And we saw that what we were doing was not really doing modeling so much as syntactically unifying existing standards. I mean, there's only 50 states in the United States and they already have abbreviations like CA and NY and stuff. Let's call that our set of state codes, right? So we brought in all of these little pieces and said these are the building blocks, come use them, and you are welcome to come use these things and come join our club if you ean contribute technically.

Mr. Ellertson: I would add one final comment, which is there seems to be a little bit of a view that some of the XML technology isn't being utilized to do or being built and it's way off in the future. I think that's a misnomer. We're building it with a number of Fortune 1000 and even smaller companies to do in real live applications like those that have been talked about to do, and we shouldn't give you too much of the impression that that's five years off because it's not, that's happening today. In addition to that, there are some tools. There are consortiums out there which have been reviewed in the different presentations today that is a starting point for common DTDs, semiconductors, the OFX for financial, and those are relatively stable and they're being used for different purposes, and if your type of information eonforms, then you're going to find some communality across that. And then, in addition, there are tools for things like DTD design, graphical DTD design like a near and far product, one of our partners, which is the graphical builder in our tool, and it allows you to graphically lay out a DTD like you would an object model or a

relational database and build it on that basis, so there are tools out there to do built for this that generate XML and there are standards, bodies, and groups that have already gotten together in various industries, and I would start there in addition, if you have to do something now real time.

Mr. Glushko: Thinking is hard, developing a DTD is hard thinking.

Ms. LaPlante: Okay, yes, in the back.

Audience: Art Harrison with the Adplex Companies. How would you classify XML? Would you say it's a subset, a superset, or a complete replacement for HTML? And secondly, how would you say that relates to, say, active server pages for tying into a database?

Mr. Ellertson: Well, I'll take a part of it and let these guys take some of it too. But XML clearly allows you to separate structure and style, and those are imbedded in HTML, so I can't do that. So when I have my content or my information broken up into those tagged elements, as we've seen in all of the presentations, I have the ability now to use a template or a style language, with the DTD extract certain pieces of information, and then apply a style for an output, the font size, the form of the paper, that type of thing. I cannot do that with HTML, I can't even do it really with dynamic HTML. So it adds intelligence contextually and then the ability to break up the structure and style.

Audience: So you're saying that it is an embeddable subset of HTML, something that adds to HTML instead of an HTML document, or do you replace the entire structure behind HTML?

Mr. Glushko: Some of this is a stylistic issue. I happen to think that I'm going to have XML documents, and I have a little bit of HTML documents somewhere else, especially things that are coming out of structured sources. If I have a database or a spreadsheet or an application which can spit out a structured data stream, I'm going to preserve that information in an XML mark-up language. I mean, taking any approach with turning into HTML for rendering and distribution is like giving it a lobotomy. You can basically want to preserve as much intelligence as long as possible, but I'm not going to tell my sixth graders, who have their one-page home pages, to learn XML. It's just not worth the effort.

Mr. Ellertson: Another way to look at it would be that SGML is maybe the most complex and has the most information, XML is a subset, and HTML is a further subset, and one of the benefits, as an example in my presentation, those screens were all generated through dynamic HTML but the data's XML, and since HTML is a subset, I can generate all of that. So beeause you don't have a browser to do in every case going back because some browsers are much older than the current that renders XML, I would want to render HTML out of

my XML-based system.

Mr. Severson: And let me give you a simple but precise kind of technical explanation to your question. SGML gives you the means of describing various kinds of applications or tag sets, different kinds of applications full of objects that you can describe. One such application is HTML, which was meant as a one-size-fits-all kind of formatting application and then later on interactive forms and so forth, so it's an application of SGML. But SGML itself allows you to describe many kinds of applications. XML is a strict subset of SGML. It is also a way of describing many kinds of applications but with a much of extra features removed, so it's kind of right down to the bare bones and gets to the point. In terms of active server pages, which give you the ability to execute scripts on the server that then dynamically ereate HTML pages, although I'm not aware of that technology being available quite yet, there's no reason on earth why that couldn't be dynamically generating XML pages in the same way.

Audience: Thank you very much.

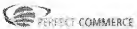
Ms. LaPlante: Thanks. Thank you all for staying. Please join me in thanking our three great speakers this afternoon. [Applause]

[End of Session]

Exhibit L. xCBL.org, About xCBL (copyright 2000) accessed at http://www.xcbl.org/about.shtml, on July 20, 2007, 2 pp.

ABOUT xCBL

ENDORSED BY



The XML Common Business Library (xCBL) is a set of XML building blocks and a document framework that allows the creation of robust, reusable, XML documents to facilitate global trading. It essentially serves one language that all participants in e-commerce can understand. This interoperability allows businesses everywhere to easily exchange documents for e-commerce, giving global access to buyers, suppliers, and providers of business services.

Related Resources

- W3C
- ebXML
- XML.ORG
- OASIS
- UBL

xCBL 4.0, the latest version, provides a smooth migration path from EDI-based commerce because of its origins in EDI semantics. xCBL will be able to support all essential documents and transactions for global e-commerce including multi-company supply chain automation, direct and indirect procurement, planning, auctions, and invoicing and payment in an international multi-currency environment. xCBL 4.0 is the first version that uses XSDL as the canonical form. Previous versions all used SOX.

xCBL is the result of extensive collaboration between Commerce One® and the leading XML standards bodies, e-commerce enterprises, and hardware and software vendors, as well as analysis of existing e-commerce standards including Electronic Data Interchange (EDI) and RosettaNet, Industry leaders Compaq, Microsoft, SAPMarkets, and Sun Microsystems are leveraging xCBL 3.0 and xCBL 3.5 as a key standard in the development and delivery of business-to-business solutions.

The Evolution of xCBL

xCBL began its life at Veo Systems in 1997. At that time it was called simply CBL, and was a research project partly funded by the Department of Commerce's Advanced Technology Program. CBL was developed to test the limits of XML for e-commerce and to identify requirements for XML design, development, and transaction tools and platforms. Subsequently, Veo invented the first object-oriented XML schema language; SOX, the Schema for Object-Oriented XML; as a result of the lessons learned in the first version of CBL.

In January of 1999, Commerce One acquired Veo Systems and the CBL technology. Commerce One saw the vision of conducting business electronically using XML that it embodied would help complete its transformation from an e-procurement company to a business internet company, and indeed, would transform the concept of e-commerce. The Veo CBL was tailored to support the Commerce One products and customers, which entailed making it interoperable with EDI (Electronic Data Interchange). This led to the creation of xCBL 2.0, and at that time the "x" was added to reflect the relationship with XML. xCBL 2.0 added a strong non-proprietary

R-716

and interoperable semantic foundation for CBL, and gave companies using EDI a way to transform those applications to XML.

xCBL 3.0, released in December of 2000, represented a major broadening of scope and is more powerful than xCBL 2.0. It is rich enough to encode any e-commerce standard, and allows users to build customized documents from standard components. xCBL 3.0 saves developers and integrators enormous amounts of time and effort, and ensures that recipients are able to understand all documents that they receive when doing e-commerce. With version 3.5 of xCBL, support for the W3C Schema (full recommendation) is provided. This means that you could now choose between Commerce One's SOX, the W3C XSDL, Microsoft's XDR as well as the original DTDs when developing solutions that use xCBL. Only minimal changes to the document instance are required; please see the following link for more information *xCBL User's Group Newsletter*. With the latest release, xCBL 4.0, XSDL is used as the schema canonical form and some initial alignment with standards defined by UBL .

The Future of xCBL

UBL was announced on October 17, 2001 stating that xCBL will be the starting point. Specifically the UBL Charter states that the UBL Technical Committee will ... *"Begin with xCBL 3.0 as the starting point ... to develop the standard UBL library by mutually agreed-upon changes to xCBL 3.0 based on industry experience with other XML business libraries and with similar technologies such as Electronic Data Interchange."* In many ways, the results of UBL are likely to be similar to xCBL but based on the input of many more individuals. Given the progress of UBL where does this leave xCBL and how, in the future, will xCBL relate to UBL?

In Feb, 2003, UBL released the Op70 version for public review which included the Library Content part of UBL that specifies a library of business information entities to be used in the construction of business documents. The release also included a set of common XML business documents assembled from those entities. UBL does not replace xCBL in the short term; it is in its development phase. Currently UBL does not contain the depth and breadth of components and documents that are in xCBL. As UBL continues to develop xCBL will migrate to this standard.

Many of the business information entities developed as part of the UBL initiative have equivalents in xCBL. These new UBL components will replace their xCBL equivalents in future releases of xCBL on a phased basis. In the short term, it is unlikely that there will be UBL components for all the existing xCBL components although eventually most existing xCBL components and probably many of the xCBL document structures will be completely replaced by their UBL equivalents and mapping of existing xCBL documents to UBL documents will take place. Commerce One also plans to publish mappings from xCBL to UBL to make it even easier for xCBL implementers to migrate.

R-717

Exhibit M. Allen, Common Business Library (CBL) (May 1999) accessed at http://www.infoloom.com/gcaconfs/WEB/granada99/all.HTM on July 20, 2007, 9 pp.

Graphics-based Product Documentation: Principles and an Application | Table of contents | Indexes | HL7-XML Progress Report

Common Business Library (CBL)

Terry Allen
Commerce One
Email: tallen@sonic.net

Biographical notice:

Terry Allen is a specialist in technical standards that support complex electronic publishing applications, including information discovery and retrieval, metadata, and internationalization. He is a codesigner of the Docbook DTD, the SGML application most commonly used for computer documentation (and good for other things, too!). He has participated in IETF and W3C working and special interest groups on HTML, URLs, URNs, MIMESGML, WEBDAV, XML, and the OCLC Metadata group. He designed and edited the first Web portal site (Global Network Navigator's Whole Internet Catalogue). Since 1997 he has been working on document design and architecture for electronic commerce systems. He is currently chairman of the OASIS Registry and Repository Technical Committee. For further details seehttp://www.sonic.net/~tallen/

ABSTRACT:

In this paper I'll described the XML e-commerce language I wrote in 1997 and 1998, its purpose, design goals, architecture, and features. I'll describe the semantics I built upon and discuss lessons learned from the project, including the phase in which the original DTD syntax was transformed into an XML schema syntax. And I'll indicate directions for future development. CBL's development was partly funded by the U.S. Department of Commerce i (NIST) Advanced Technology Program award 70NANB7H3048 to Veo Systems, CommerceNet, BusinessBots, and Tesserae Information Systems.

Copyright 1999 by Commerce One, Inc.

Why CBL?

CBL (Common Business Library) , was developed as an experimental prototype e-commerce language in XML (Extensible Markup Language) to describe

documents exchanged among components of a componentized e-commerce system. The notion that an e-commerce system should consist of components is the "ECO" strategy enunciated by Marty Tenenbaum, former Chairman of Veo Systems (which was acquired by Commerce One earlier this year). In such a system services offered by components can be assembled into larger and varied services. The advantage of using XML documents as interfaces instead of CORBA (Common Object Request Broker Architecture) objects (as was envisioned originally) is that they are easier to develop and use for a wider range of users, and that they represent naturally the documents, whether paper or EDI (Electronic Data Interchange) , already in use in commerce. It's important to understand that CBL is an artificial language for e-commerce and e-commerce documents, not an exercise in artificial intelligence.

CBL includes not only the XML DTDs and SOX (Schema for Object-Oriented XML) schemas, along with their associated documentation, but also a specification for constructing compound XML documents and packing them in MIME (Multipurpose Internet Mail Extensions) Multipart/Related messages and to a certain degree a model of document interchange (SOX is Commerce One's submission to the W3C (Worldwide Web Consortium) for consideration by the XML Schema Working Group).

I began work on CBL in August 1997. Version 1.1 was released in September 1998, and version 1.2, which is represented in both DTD (Document Type Definition) syntax and SOX , was completed in November 1998. At that point the specification had done its job in providing proof of concept, both to us and to the many who have downloaded the distribution; CBL is currently being employed only as a reference model. Further work is desireable to harmonize CBL semantics with those of EDI (both the X12 and EDIFACT flavors), and with other specifications that have appeared since my CBL work began. In the meantime, we've used a simplified subset of CBL for several successful demo projects.

Design Goals

For the most part CBL DTDs represent familiar document types, such as catalogue entries and purchase orders. These document types, and the mechanisms used to construct them, originally were chosen in order to meet the following goals:

- Cover a broad range of common requirements at a reasonably simple level while designing for extensibility.
- Use existing implemented and stable functionality insofar as possible, by using international and IETF (Internet Engineering Task Force) standards and proposed standards where practical and to the extent they are useable, while steering clear of contentious, unimplemented, unstable, or unready specifications or standards. XML , to the degree it is used in CBL , is considered a stable application profile of SGML (Standard Generalized Markup Language) , an international standard for which there exists considerable record of successful use. (Later, I created a SOX

- representation of CBL , as I'll explain shortly.)
- Leverage reusability of documents and markup constructs.
- Make internationalization and localization of documents easy.
- Provide sufficient data typing to enable the construction of programs to process CBL documents.
- Maintain independence of transport protocol.
- Make it easy for electronic commerce participants to establish trust and exhibit good faith.

Starting Points

My first step was to scour the Web for e-commerce standards and specifications. There is no lack of them, but a year ago August there was very little in XML or in a format lending itself to XML ification. I found a number of obvious standards for the basis of an e-commerce specifications, such as ISO 8601 for date and time, and ISO 4217 for language codes. I also found the BSR (Basic Semantic Repository) , which contains a partial union set of semantics from X12 and EDIFACT. From the BSR I extracted primitives for such things as addresses. (You can find the BSR online at.)http://www.iso.ch/BSR/

I then examined specifications for sets of e-commerce documents, such as OBI (Open Buying on the Internet) , and in concert with my colleagues at Veo, devised a set of document types that would support both the construction of online trading communities and the scenarios of such specifications as OBI . I aligned relevant document types with semantic primitives defined by Rosetta Net (for catalog content) and the IOTP (Internet Open Trading Protocol) (for payment). Then I worked through as many business semantics models as I could to see if my document types were sufficiently robust to do real work, and tested them by constructing sample documents to support various specifications such as OBI .

CBL Architecture

From the standpoint of DTD design, CBL 's DTD-syntax representation is traditional: it has an information pool composed of modules, some of which rely upon each other, and a set of modules that define the contents of document types; the document types themselves are meant to be containers that can be discarded when building larger document types.

As part of CBL I

- developed further the MIME Multipart/Related compound document packaging proposal I've been promoting for several years (see my SGML '96 paper, "Package or Perish"),

R-721

- adopted the general syntax of the original XLink proposal, with some embellishments,

- worked out the usage of URNs in CBL such that resolution over the Net would rarely be required. Every CBL document type may contain a metadata section at its outset. The metadata elements provided to date are minimal, but they include, optionally, one or more URNs. A URN (Uniform Resource Name) , which is uniquely a name for the document, can be used later in an exchange of e-commerce documents as a proxy for the document itself, without requiring the construction of a URN resolution service, provided its name space does not allow the assignment of URNs for variable resources such as "the latest version of something." If you have already acquired a copy of a document (such as a taxonomy) identified by URN , when it is pointed to by URN in a new compound document (such as a product description), you can assume safely that you need not refetch and examine it. In general, CBL is constructed so that you can use URLs everywhere you can use URNs, and so that you can use URNs in typed pointers as shorthand for a piece of a compound document.

- made extensive use of typed pointers (pointing constructs that must target a document of a particular document type). Large-scale documents are constructed of sets of smaller ones in part by use of these pointers, thus retaining the semantics of document types as they are aggregated and allowing for lazy evaluation and easy semantic integration of disparate document types. A typed pointer that points to a contact information document looks like this:

```
<market.participant.info.pointer>
<urn.reference
urn.string="urn:x-commerceone:identity:henry.morgan">
</urn.reference>
</market.participant.info.pointer>
```

- These link elements bear an attribute, fixed in the attribute declaration, indicating whether the documents they point to are properly content of the parent document or lie outside of it. This device allows one to draw a limit around the logical content of a compound document (or, more generally, a hypertext collection) and avoid unwanted recursion in processing. CBL's link element syntax is compatible with the provisional XLink specification.

- constructed a taxonomy DTD for description of products along various axes. This taxonomy model describes a recursive hierarchy of taxons, in which taxons contain child taxons. In order to avoid growing a single enormous tree, taxons may point to parents, children, or siblings in other taxonomies. Taxons have names and descriptions, and may have sets of keywords. They may also contain descriptions of queries that may be made against schemas describing instances of the entity classified by the taxon: this device enables a

- user to know what questions it makes sense to ask about a schema for a particular product.
- built a query language for XML documents, including provision for querying across document boundaries within compound documents. This query mechanism is based on the Xpointer mechanism in the XLink specification. CBL uses only part of the Xpointer semantics in its query mechanism, representing them in instance syntax and augmenting them to support logical AND and OR, and traversal of links. This mechanism is considered a placeholder for an XML query mechanism yet to be specified.

E-commerce requirements not dealt with in CBL or dealt with only minimally include:

- workflow and service description
- updating and versioning (it is intended to use the IETF WEBDAV (Web Distributed Authoring and Versioning) specification)
- security and digital signatures (although there is a placeholder for digital signatures in the MIME packaging specification)
- message acknowledgements
- legal information including terms of business
- payment (work remains to be done on aligning CBL with IOTP)

Semantic Domains

XML markup, in association with its documentation, gives meaning to a document's content. (Markup is often falsely called "self-describing metadata," but of course the description is in the markup documentation, not the XML document.) For e-commerce, this meaning covers three basic domains:

- Business document semantics ("line item")
- Product description semantics ("has four-wheel drive")
- Business logic semantics ("extend no credit to new customers")

Common to all these domains are at least some from among the general datatypes describing such things as time, space, number, and physical properties. Aside from that commonality, these domains are largely disjunct, but software that processes e-commerce information must deal with them jointly:

- business logic is applied to business document semantics
- business documents must handle product semantics

- product descriptions must eventually support manufacturing logic, which will be generically similar to business logic

In CBL I have dealt with these domains in different degrees:

- SOX and CBL in SOX provide some datatypes.
- CBL provides a set of business document semantics designed to support traditional SGML goals, including information reuse and also object-oriented code generation.
- I have left specific product description semantics undescribed except for a demonstration; CBL provides for general product description semantics. There is an ocean of specific product semantics (such as those now being specified by Rosetta Net), which can also be used to support manufacturing specifications, but a general e-commerce language should not include that entire ocean. I've provided a slot for specific product descriptions within the product description and catalogue entry document types, so any specific product description may be used.
- I have considered business logic out of scope for CBL —I have provided only a placeholder pointer to it in the MIME compound document specification. I may send you a purchase order so you can fulfill it, notarize it, or archive it, and I have to indicate my intention outside of the document if it is to be used unchanged for all those purposes.

Transition to SOX

About the time I began to have things worked out, Murray Maloney, Alex Milowski, and Matthew Fuchs began to develop what became SOX , and I spent many hours converting my CBL DTDs into SOX schemas, initially in a mechanistic fashion, later taking into account SOX's mechanisms of inheritance and extension. By version 1.2 of CBL I was writing SOX schemas first and thinking in SOX ; the DTDs became secondary products. This development led to considerable discussion within the company about how to use SOX effectively—and my education in the alternate universe of object-oriented programmers. But it is from SOX source (more properly, source conforming to a rearranged subset of SOX) that our programmers have been working for the past year.

Going forward, we intend to use the XML schema language that the W3C will specify.

CBL Lessons Learned

The Obvious

R-724

We can't hand-craft everything . It will be necessary to generate huge wads of SOX from existing specifications. Optimization probably must be done by algorithm or not at all, rather than in Terry's wetware.

Naming is contentious . As you all know. Our Java programmers complained unendingly about my naming syntax (lower case with periods as separators). I have since discovered a multitude of naming styles in existing e-commerce specifications that are similarly unlike Java conventions. I'm afraid programmers will have to live with syntaxes they don't like; models of multiple names associated with multiple contexts, such as ISO 11179, may be useful for relieving this tension. For example, in IEEE P1489, the Standard for Data Dictionaries for Intelligent Transportation Systems, one find a name of the form`CONSTRUCTION.ROAD_TargetCompletion_date` . In an ISO 11179 schema one might represent this as the name of the data element and add

```
<synonymous.name
context="Java">constructionRoadTargetCompletionDate
</synonymous.name>
```

I did not try to avoid qualified names for element types and attributes, but I did try to avoid the flattened names of EDI . For example, where CBL has an`address` element EDI has compounds such as `Goods.DeliveryLocation.Address` , which collapse containing context with the names of elements that ought to be reuseable. These collapsed constructs are not unlike queries, but they have no place in an XML schema.

Heritage

EDI transaction sets are useful . At least some of them, anyway. They represent information sets people actually want to exchange, and can be transformed into document types by deleting unneeded information (such as trailer fields) and separating out the semantics of individual documents from those describing batches of documents.. Rosetta Net is developing information interchange models that includes stubs for what could be document types (Rosetta Net is current filling in these stubs with EDI transaction sets), and document types developed by the Open Applications Group look like a rationalized version of EDI's. There is little point in reinventing these information sets, although new ones are required for describing markets, their workings, and their participants.

Certain EDI basic semantics are useful, but BSR is not the solution because of the collapsed context problem I mentioned earlier. X12 and EDIFACT badly need reengineering and rationalization, so as to sort out reuseable primitives (Address) from contextual semantics (CustomerAddress). And above all their archaic syntax has to be junked.

OO XML

Multiple inheritance is essential . Corky (who exists only for the purpose of this example) is a swine, a sow, a mother, a*Sus scrofas domesticus* , a Gloucester Old Spot, a pet, a thing licensed by Sonoma County, a patient of Dr. Clive N. Huff, the object of an insurance policy, and an input into the manufacturing process that will produce this fall's supply of pork. There is no way to arrange the matrix of Corky's attributes in a single tree. The best we can do is construct a SOX representation of a swine schema that uses inheritance along the axis most useful to our immediate needs, and represent information along the other axes by XML containment and pointers to taxonomies (taxonomies are very important).

SOX -based e-commerce schemas require constructs not present in EDI . For example, in CBL 1.2 I have both a`simple.line.item` and an `unpriced.line.item` (for those cases such as a request for bid, in which the price is unknown and should not appear). For SOX I've created quite a few prototypes (38 at last count) to support inheritance, which EDI knows nothing of.

A lot can be done with datatypes . Most enumerations can be reduced to datatypes. The ability to specialize datatypes and provide constraints on acceptable values is very powerful.

The Not Obvious

You can be too abstract . At one time I modelled a general transaction description document type. It helped me a lot in realizing what a transaction is (the exchange of value between entities, perhaps many such exchanges) but it was too abstract to be useful in the real world. So I broke it down into purchase order, invoice, request for bid, response to request for bid, which is the level of abstraction found in EDI and the level that people find comfortable.

Semantic mapping is essential . To use semantics already defined elsewhere, it is necessary to be able to point to them, a facility added to SOX in its latest revision. It is worth noting that ISO 11179, which deals with the specification of data elements and the organization and operation of a data element registry, has facilities for doing semantic mapping (and has considerable intellectual overlap with SOX). Whether this mapping should be done within the XML schema or from outside, in an independent document, depends, I think, on whether one is trying to reuse well known definitions of semantics in a new schema or construct mappings among existing schemas (which may be read-only).

Business logic must be expressed . To construct efficient schemas for business documents it is necessary to know how the information they encode is to be processed—or at least what sets of information will be processed together. I've considered it out of scope for CBL , but I'm beginning to wonder if I'm right about that.

Process logic must be expressed . I may send you a purchase order for you to fulfill it, for you to notarized it, or for you to archive it. I can't express that intent within the document, which I may want to digitally sign and use unchanged for all three

R-726

purposes. I found I couldn't reasonably express processes in CBL ; something along the lines of UML (Unified Modeling Language) is needed and I now just point out to a hypothetical process description. I'm pretty sure this is out of scope for CBL , but it's needed for a complete e-commerce system.

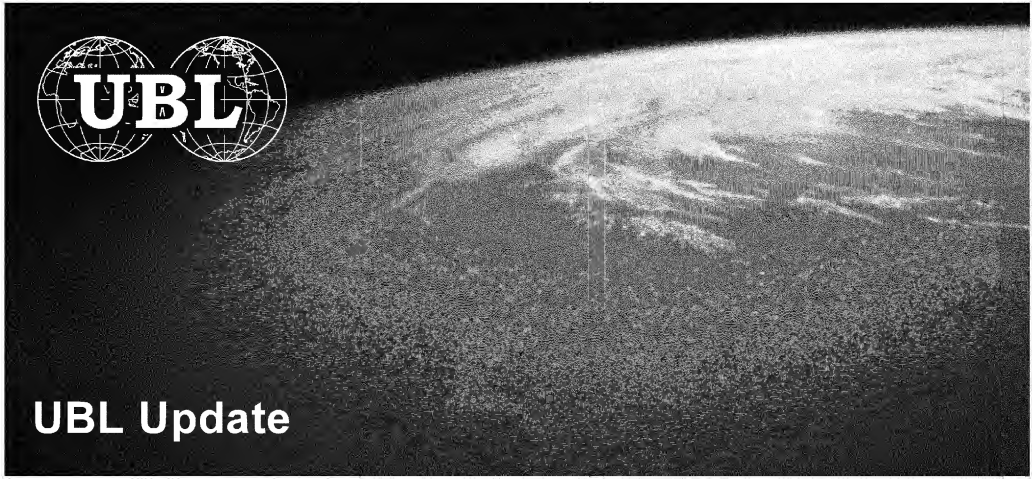
Both registries and repositories are essential . ISO 11179 defines a registry, which can be seen as an interface to a repository. There is some variability in what these entities are called by different people, but the distinction is between metadata (the information held in the registry) and the data (the DTDs or XML schemas themselves). For XML to succeed on the Web we need a means of serving DTDs on demand. To enable sane development of XML schemas, we need to enable reuse of XML schema source. In both cases a repository is required: for XML on the Web, it is what an XML client would access (perhaps directly, without going through a registry) to resolve a DOCTYPE declaration. For an XML development environment, a registry is essential to permit an overview of what exists already and can be reused, and the repository behind the registry is essential for managing authoritative source (I managed the registry part of this problem in wetware during CBL development, but even inside my head that approach doesn't scale).

Beyond CBL

CBL was an interested exercise as a prototype, and allowed me to develop solutions to many XML architecture problems. But from a practical point of view, its semantics are insufficiently integrated with those already used (in a various and confusing ways) in commerce.

We intend to respecify CBL in the ECO Framework Project forum (described at), embracing as much of the semantics of EDI as possible. To do this we must determine what pieces of EDI semantics are actually used and used consistently, we must create uniform XML representations of EDI's various code sets (and the code sets it relies on, such the list of all the world's airport codes), and we must rebuild EDI's innards. As I remarked earlier, EDI's document types are useful—so they are a starting point for top-down design. And the atomic data elements are useful—so they are a starting point for bottom-up design. In the middle there's the problem of what comes in the middle. Some of EDI's "segments" are probably useful; larger structures common across document types need to be defined. And, as I discovered in the course of conversion to SOX , one needs prototypes to provide a common basis for structures that share some contents. A respecified CBL will be much richer in both prototypes and large structures than EDI — representing XML's facilities for reusing information.http://www.commerce.net/projects/currentprojects/eco/wg/

Exhibit N. Bosak, UBL Update, OASIS Symposium on the Future of XML Vocabularies, Slide 3 (Apr. 25, 2005) viewed at www.oasis-open.org/events/symposium_2005/slides/bosak.pdf on July 20, 2007, 10 pp.



UBL Update

Jon Bosak
Sun Microsystems

OASIS Symposium on the Future of XML Vocabularies

New Orleans, 25 April 2005

http://oasis-open.org/committees/ubl

Sun microsystems

OASIS

The Universal Business Language (UBL)

- International effort to define a royalty-free library of standard electronic business documents
- Designed in an open and accountable vendor-neutral OASIS Technical Committee with participation from a variety of industry data standards organizations
- Plugs directly into existing business, legal, auditing, and records management practices with minimum disruption
- Eliminates re-keying of data in existing fax- and paper-based supply chains
- Fills the "payload" slot in B2B web services frameworks
- Maintains close alignment with existing EDI systems
- Presents vendors with a standard target for cheap off-the-shelf business software

UBL 1.0: the "Fifth Generation" B2B language

G1 (1Q 1998): CBL 1.0 (Veo/NIST)

G2 (2Q 1999): CBL 2.0 (Commerce One)

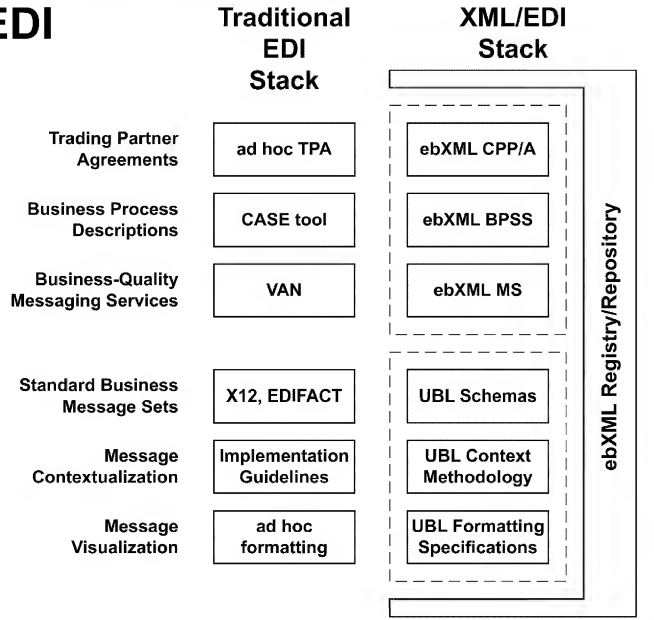
G3 (4Q 2000): xCBL 3.0 (Commerce One and SAP)

G4 (1Q 2003): UBL 0.7 (OASIS)

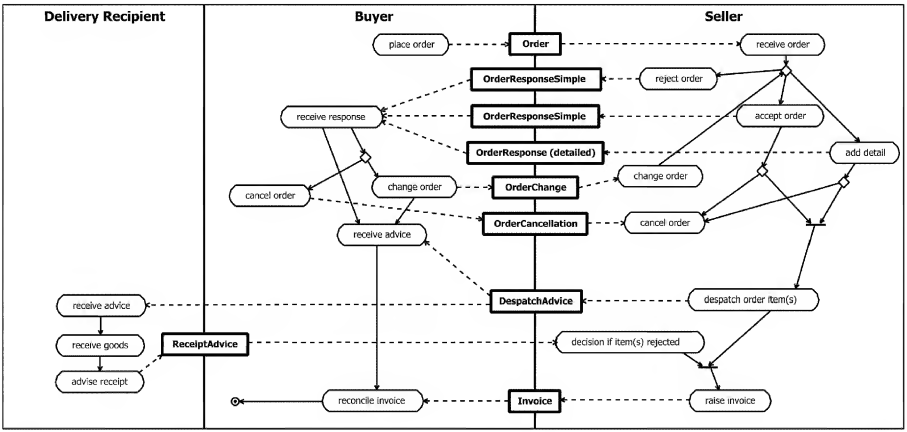
G5 (4Q 2004): UBL 1.0 (OASIS)

UBL represents over six years of continuous development in the creation of a standard XML business syntax.

XML EDI


Traditional EDI Stack
XML/EDI Stack
Trading Partner Agreements
ad hoc TPA
ebXML CPP/A
Business Process Descriptions
CASE tool
ebXML BPSS
Business-Quality Messaging Services
VAN
ebXML MS
Standard Business Message Sets
X12, EDIFACT
UBL Schemas
Message Contextualization
Implementation Guidelines
UBL Context Methodology
Message Visualization
ad hoc formatting
UBL Formatting Specifications
ebXML Registry/Repository

UBL order-to-invoice



This model describes a very large class of use cases.

Recent developments

- UBL 1.0 ratified as **OASIS Standard** (November 2004)
- UBL 1.0 Naming and Design Rules ratified as **OASIS Standard** (January 2005)
- UBL 1.0 International Data Dictionary released as an **OASIS Committee Draft** (April 2005), enables deployment in Chinese, Japanese, Korean, and Spanish
- Free stylesheets and UBL formatter provide bridge to paper form
- UBL Naming and Design Rules (NDR) adopted by major industries: chemicals (CIDX), petroleum (PIDX), agriculture (RAPID), real estate (OSCRE/PISCES), U.S. Department of the Navy (DON)
- UBL Invoice required for all public sector invoices in Denmark (February 2005)
  - Estimated annual savings 94 million Euros (117 million USD)
  - If the UBL Purchase Order is implemented, annual savings are estimated at 160 million Euros (199 million USD)
  - Over one million invoices exchanged in first two months of operation
- HISC and SBSC formed to develop Small Business Subset
  - Provides standard input forms and "starter kit" for inclusion in XML products

Next step: UBL 1.1

Work now continues on an update due out by the end of 2005.

UBL 1.1 will include:

- Resolution of a number of issues raised during the development of UBL 1.0 that were judged safely deferrable to 1.1
- Refinements and clarifications of the UBL data definitions arising from the translation work
- Additions to the UBL library identified by the UBL localization subcommittees as required to accommodate regional business practices; example: ECALGA (Japan)
- Additions to the UBL document set submitted by qualified organizations (Example: Certificate of Origin)
- Additional support for taxation requirements from the OASIS Tax XML TC
- Extensions to the UBL procurement model proposed by IDA (EC) and OGC (UK) to implement a common European government eprocurement process; result will provide document types for pre-ordering phase (catalogue, request for quotation, quotation) and post-ordering phase (credit note, debit note, statement, remittance advice)

Challenges

- All the usual stuff (getting people interested, keeping on schedule, etc.)
  - Hard to see how this could succeed without an (almost) full-time chair
- Lack of commitment to the majority rule decision-making process
- Secretarial support (site maintenance, calendar, issues tracking...)
  - Root of the problem: it takes a subject-matter expert to do this properly
- Versioning and naming
  - No coherent OASIS process that addresses all the hard bits
  - Takes a lot of thought that no one has time for
  - Inability to construct a sensible file tree makes this harder than it has to be
  - Possible solution: OASIS Librarian
- Meeting logistics in an international context (see next slide)

International meeting logistics

- F2F meetings will become increasingly difficult as travel becomes more expensive; this model of standards work probably doesn't have more than another 2-3 years
- Solution #1: phone conferences
  - Problem: scheduling
    - Solution: Split meetings
      - Atlantic call: 08h00 in San Francisco, 17h00 in Brussels
      - Pacific call: 16h30 in San Francisco, 08h30 in Beijing
    - Cost: Communication and coordination overhead
    - Short-term ad hoc work groups can help with this
  - Problem: audibility
    - Solution: Wireless microphones
    - Challenge: No community adoption, no standards
    - Proposal: An OASIS TC to address this
- Solution #2: regional work groups
  - SC membership independent of TC membership will be a big help
  - We need to develop community norms around this approach

For more information

UBL: http://oasis-open.org

UBL 1.0: http://docs.oasis-open.org/ubl/cd-UBL-1.0.zip

UBL 1.0 Naming and Design Rules: http://www.oasis-open.org/committees/download.php/9943/cd-UBL-NDR-1.0Rev1b.pdf

UBL 1.0 International Data Dictionary: http://www.oasis-open.org/committees/download.php/12205/cd-UBL-1.0-IDD-1.xsc

UBL UN Layout Key stylesheets: http://www.cranesoftwrights.com/u/

UBL/Layout Key Transformer: http://www.ambrosoft.com/

ubl-dev: http://www.oasis-open.org/mlmanage/

ebXML: http://ebxml.org

freebXML: http://www.freebxml.org

OpenOffice: http://openoffice.org

xmlroff: http://xmlroff.sourceforge.net/

UBL chair: jon.bosak@sun.com

Exhibit O. Meltzer and Glushko, XML and Electronic Commerce: Enabling the Network Economy, SIGMOND Record, Vol. 27, No. 4, 21-24 (Dec. 1998), 4 pp.

XML and Electronic Commerce: Enabling the Network Economy

Bart Meltzer and Robert Glushko
VEO Systems
4005 Miranda Avenue, Suite 150, Polo Alto, CA 94304.
email: info@veosystems.com

There has been a lot of talk about how the Internet is going to change the world economy. Companies will come together in a "plug and play" fashion to form trading partner networks. Virtual companies will be established and new business models can be created based on access to information and agents that can carry it around the world using computer networks.

We totally believe in this grand vision and are confident that it will begin to happen in the next few years. We are also realistic and understand that there are still many barriers to enabling the vision. EDI, although a success for some, has not been accepted by the majority of the business community as a way to do business electronically. Even though there have been extensive efforts to standardize EDI transactions, it remains too expensive and the software developed does not make it easy to leverage implementations across different trading partners.

These barriers to adoption of EDI leave many businesses with paper-intensive, manual, costly processes to exchange business documents, forms and messages with their trading partners. Neither EDI nor a manual paper process can change at the same pace as the dynamics of a business. Any solution that is targeted at enabling the network economy must be one that insulates businesses operating computer systems from the daily changes that occur in a business.

For this reason, our approach to enabling the network economy is to make the business documents, forms and messages that flow between businesses, comprehensible to each business no matter what computer system is used and even if each business is using different computer systems. XML used in Electronic Commerce is an enabling technology that makes it possible for business documents forms and messages to be interoperable and comprehensible. XML is one of the key ingredients that will accelerate the reality of a network economy and new business models based on the Internet. Early work with XML and electronic commerce is happening in the area of procurement, distribution, supply chain management.

So why is XML a good enabling technology?

The features of XML are:

- A markup specification for creating self descriptive data
- A platform and application independent data format
- A way to validate the structure of data
- A syntax that can be understood by computers and humans
- An incremental way to advance web applications used for electronic commerce

The benefits of XML to business are:

- Businesses can describe services in a manner that can be widely understood
- One set of documents, forms and messages can be exchanged by businesses with different internal business systems
- Errors in re-keying data are reduced because data can be transformed through gateways
- Frequent changes in business process can be handled without substantial engineering cost
- Leverages investment in legacy systems and can be used with latest Internet technology

To further understand the benefits of XML, it is important to compare it with today's use of HTML. HTML, a simple markup language for display, has limited facilities for distinguishing between content, structure, style or relationships. In order for the existing electronic commerce applications to function, many proprietary tag sets and extensions to HTML have been developed. However, the resulting lack of standards makes the content with these applications incapable of inter operating with the applications of trading partners.

In the following HTML description of a computer it is not possible to distinguish the structure of the description from the way it is displayed. Thus there is no way to mark up important information in a meaningful way for use by an application.

```
<TITLE>Laptop Computer</TITLE>
<BODY>
<UL>
<LI> IBM Thinkpad 560X
<LI> 233 MHz
<LI> 32 Mb
<LI> 4 GB
<LI> 4.1 pounds
<LI> $3200
</UL>
</BODY>
```

For example, if a catalog search application (or application component) got the input:

"Find laptops with at least a 200Mz processor, more than 800K of disk space, and less than 300,000 yen" the search would not be able to return this HTML page unless there was rigid (and expensive) parsing code for the content of the page. Even though the laptop meets all the requirements, the markup isn't "smart enough" to support the transformations of data units needed to match the query.

In contrast, the same computer description in XML provides a much more usable description of the same information.

```
<COMPUTER CLASS="Portable">
<MANUFACTURER>IBM</MANUFACTURER
>
<FAMILY>Laptop</FAMILY >
<LINE>Thinkpad</LINE>
<MODEL>560X</MODEL>
<SPEED UOM="MHz">233</SPEED>
<DISK UOM="GB" >4</DISK>
<WEIGHT                    UOM="lb"
STANDARD="ISO">4.1</WEIGHT>
<PRICE CURRENCY="USD">3200</PRICE>
</COMPUTER>
```

In this case, is easy to understand each atom of the document thereby making intelligent processing of the document possible. Processing "XML anywhere" is a more achievable statement than what Java will ever be able to produce. Because XML is declarative data, processing issues are not as complex as they are with running Java everywhere. An example to processing XML anywhere can be derived from the previous computer description.

- The schema for computers can be validated by one of the XML processors on the market (some for free). Validating XML processors are becoming more widely available as XML gains market momentum.
- The XML tags provide a logical container for extracting and manipulating <COMPUTER> information as in the case of a sorting algorithm associated with a catalog search.
- It is possible to sort by <MANUFACTURER>, <FAMILY>, <LINE>, <SPEED>, <WEIGHT>, <PRICE>, etc.
- There is an explicit identification of each part that enables its automated processing such as converting <PRICE> from "USD" to "Yen."

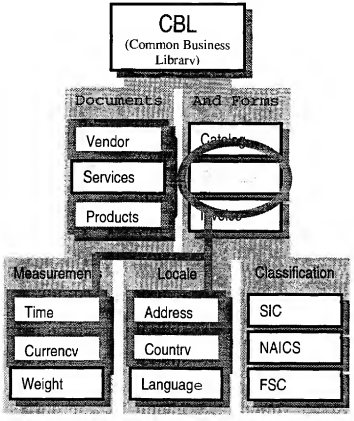
Not only does XML make it easier for computer to process information, it is good for humans and can be used to produce the same "web for eyeballs" that exists today and will continue to exist into the future. XML and XSL (a standard XML Style Sheet Language Specification under development) will make it very easy for web servers to serve XML document and XSL style sheets to browsers that will actually be transformed to HTML, DTHML and Java script when rendered within the browser environment. The following diagram is a representation of the "web for eyeballs."

The fact that XML is good for humans and good for computers can not be overlooked. Going back to EDI, a major complaint from experts is that the disconnected systems are the result of errors. Reduction of errors has a direct impact on the company's bottom line. Therefore routing information to humans for decisions as part of a business processes is a desirable concept. EDI makes this difficult because there is not enough information in the transaction sets for a human to understand without a thick binder of implementation documentation to accompany the transaction sets. Using an XML markup representation of the EDI transaction documents, it is possible for interactive web application to be woven into the existing EDI processes. For example, an XML system integrated with EDI would make it possible for an EDI document to be posted on web sites for browsers, passed around between disparate ERP systems, e-mailed to humans for review and editing, and become part of other EDI transaction documents. There are already efforts underway through Commerce Net and the XML/EDI working group to create standard tag sets for the EDI X.12 transaction documents.

XML enables anyone to develop tag sets and definitions for a domain-specific language based on a standard. This flexibility is a wonderful thing. However, this feature of XML can become a liability if not managed. If every company were to develop its own language, there would be no interoperability and years of hard work and understanding from the EDI standardization process would have been thrown away. This knowledge is too valuable to waste, so we are attempting to engage the electronic commerce community at large in the XML standardization process. The goal is to create the most robust, open framework for electronic commerce based on XML. So far this has been the effort of Veo Systems Inc., of Mountain View California, to create a Common Business Library of XML components for use in electronic commerce applications.

Every business should NOT have to figure out how to use XML to express universal concepts like "address" to include in all of its documents that require an address. Through standards organizations such as ISO and ANSI, the world has agreed-upon ways of expressing concepts like address. Veo Systems, with help from an Advanced Technology Program research award from the National Institute for Standards and Technology, has expressed these standard commerce transaction documents in XML using an architecture that makes it possible to compose documents from smaller document "building blocks."

The following diagram shows a simple example of how it is possible to compose a purchase order form from smaller markup components.



Veo has offered its Common Business Library for public use, free of charge. In September 1998, CommerceNet, the leading industry association for electronic commerce (hundreds of corporate members in over twenty countries), formed a working group whose charter is to develop an open framework for electronic commerce based on CBL. The group consists of XML, electronic commerce, and standards experts from over 20 corporations and standards consortia. The group will take the baseline version (1.1) of CBL and work to establish the appropriate intersection of concepts, a standard vocabulary and set of tags for the concepts, and an architecture that works for the development of domain specific languages.

The following diagram illustrates how the Common Business Library will be used as a foundation for other commerce languages and protocols.

Over time, the Common Business Library will grow with contributions from different vertical industries. Veo Systems is committed to providing products and services that use the Common Business Library to create Internet trading communities and marketplaces that are open and interoperable.